UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Portal web de utilidades para el mantenimiento y desarrollo de un aplicativo de gestión de activos de empresa

Raúl del Saz González

Tutor: José Antonio Martínez Pla
Ponente: Juan De Lara Jaramillo

Julio de 2020

Portal web de utilidades para el mantenimiento y desarrollo de un aplicativo de gestión de activos de empresa

AUTOR: Raúl del Saz González
TUTOR: José Antonio Martínez Pla
PONENTE: Juan De Lara Jaramillo

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2020

RESUMEN

Cuando se está trabajando en un producto de software con un gran tamaño desde hace muchos años, se necesita de mecanismos que ayuden a poder gestionarlo de una forma eficiente, aunque a veces se tiende a conservar algunas metodologías poco optimas.

Este Trabajo de Fin de Grado se ha desarrollado en la empresa Mapfre Tech. En concreto, en el departamento de Desarrollo, Evolución y Mantenimiento del aplicativo principal de gestión de activos de empresa, en este caso, todo lo relacionado con la actividad de una aseguradora.

La motivación de esta propuesta es el análisis de procesos que se realizaban de manera manual y poco eficiente para poder desarrollar una herramienta que los centralice y los realice de manera automática y así poder realizar de una manera más efectiva el trabajo de gestión del software.

El portal desarrollado consiste en una serie de automatismos que son: el registro de instalaciones de manera manual o mediante servicio REST; una pasarela entre el SCM y el usuario con la incorporación de un analizador sintáctico de archivos PL/SQL; un generador de archivos formateados para la realización de métricas de calidad del software; un repositorio con las direcciones de los servidores de cada entorno y su estado; acceso al log de los entornos.

Al ser un conjunto de distintas herramientas, el trabajo ha consistido en desarrollar cada una de estas por separado para después unificarlas en un mismo sistema de acceso común a todos los empleados del departamento. Este sistema se ha desarrollado siguiendo el paradigma WAMP, un portal desarrollado en PHP utilizando una base de datos MySQL en un servidor Apache instalado en el sistema operativo Windows.

PALABRAS CLAVE

WAMP, Apache, MySQL, PL/SQL, PHP, Automatización, Plastic SCM, SQL*Plus, Entornos.

ABSTRACT

While working on a great size software product for many years, some mechanisms are needed to help manage them efficiently. However, sometimes there is a tendency to maintain suboptimal methodologies.

This bachelor thesis was developed at the company Mapfre Tech. In particular, in the department of Development, Evolution and Maintenance of the main applicative of company asset management, which is concerned with all activities related to an insurance company.

The motivation of this proposal was the analysis of processes conducted manually and inefficiently in order to develop a tool that allows their centralization and automation and thus allow an effective software managenent.

The developed portal consists of the following automations: the registry of installations through the service REST or manually; a passage between the SCM and the user, incorporating a PL/SQL syntactic file analyser; a formatted file generator to obtain software quality metrics; a repository with the addresses and status of the servers of each environment; access to the environments log.

As a set of distinct tools, the project was carried out by developing each of them individually to ultimately unify them in the same access system, which is common to every employee in the department. This system was developed according to the WAMP paradigm, a PHP portal that utilizes a MySQL database in an Apache server installed in the Windows operating system.

KEYWORDS

WAMP, Apache, MySQL, PL/SQL, PHP, Automation, Plastic SCM, SQL*Plus, Environments.

AGRADECIMIENTOS

A Miguel Ángel Muñoz por darme la oportunidad de realizar este proyecto y a Jamapla, mi tutor, por aguantarme, por haberme ayudado con el seguimiento durante el proceso y por su confianza.

A Juan de Lara, por aceptar ser mi ponente cuando parecía imposible y por su confianza depositada en mí.

A mis padres y a mi hermana por el sufrimiento de estos últimos meses.

INDICE

INDICE DE FIGURAS

1 INTRODUCCIÓN

En este primer capítulo se comentará el contexto en el que se ha realizado el proyecto, dando un repaso al departamento donde se ha desarrollado, así como la motivación, un resumen de la solución propuesta atendiendo a los objetivos, la metodología empleada y la organización de esta memoria.

1.1 Descripción del aplicativo y departamento de la empresa

El proyecto surge en el departamento de mantenimiento, desarrollo y evolución de un aplicativo de gestión de activos de empresa. El aplicativo tiene unas dimensiones enormes, por lo que se resumirá a continuación lo relacionado con la herramienta a desarrollar.

Aplicativo de gestión de activos de empresa

El software que se quiere controlar de una mejor manera consiste en un aplicativo de gestión de seguros, dentro de este se manejan gestión de pólizas, riesgos, terceros, presupuestos, etc. Está desarrollado en PL/SQL, Java y JavaScript, dividido en más de 40 repositorios alojados en Plastic SCM, cada uno relativo a distintas funcionalidades: un repositorio relacionado a terceros, otro para emisión, para tesorería, siniestros, etc. Así como repositorios específicos para distintos países, ya que es un aplicativo utilizado de forma internacional.

Esta aplicación cuenta con una serie de frameworks estándar como Spring y AngularJS, pero también con uno propio denominado Gaia, que enriquece a los estándar y crea componentes reutilizables para el desarrollo de la aplicación, para cada capa (Java/JavaScript) desarrollado por el departamento de arquitectura. A nivel frontal cliente le dota de componentes visuales y recubrimiento de alto nivel para los elementos de

Angular (controladores, servicios, etc.) y a nivel Java le ofrece servicios de alto nivel en forma de anotaciones que implementan y facilitan el flujo de datos entre las capas.

Consta de una API REST desarrollada en java, que se encarga de ofrecer los servicios a otras aplicaciones relacionadas con el negocio, de forma que se puedan acceder a datos internos o a actualizar registros sin tener que desarrollarlos exclusivamente para estas nuevas herramientas.

Mantenimiento, desarrollo y evolución

El departamento se divide en pequeños grupos que se encargan de los evolutivos y correctivos siguiendo el ciclo de vida del desarrollo de la aplicación, mediante tareas de análisis, desarrollo (descentralizado en factorías externas), mantenimiento de la base de datos, de la arquitectura de la aplicación o de la ejecución de las pruebas, así como proyectos externos que requieren de funcionalidad de la API. La evolución de la metodología empleada por el departamento partió de las formas tradicionales, con las que se construyó el primer proyecto, denominado Tronweb. Una vez fue avanzando el desarrollo, surgieron nuevas tecnologías y formas de trabajo, estas fueron adoptadas para dar más dinamismo, tanto al producto como al proceso de construcción. Se optó por crear un evolutivo de Tronweb al que se le denominó Newtron, incorporando metodologías Agile e introduciendo (actualmente) DevOps, en el cual se está trabajando actualmente.

Un aspecto que destacar en este apartado es la mención de los distintos entornos que se utilizan, estos son Desarrollo, Integración Continua (a partir de ahora IC) el cual es el comienzo de la evolución hacia la metodología DevOps, Integración, Preproducción y Producción. En términos generales, se utiliza el primer entorno para codificar y realizar pruebas unitarias, en IC se realizan subidas periódicas del código y pruebas de integración, Integración y Preproducción se utilizan para comprobar la correcta incorporación de las nuevas partes del código con las antiguas y las pruebas de usuario. El entorno de Producción corresponde al entregable de cada país.

De forma resumida, el proceso de desarrollo de una nueva funcionalidad o la modificación de alguna siguen el siguiente esquema:

1. Se analiza el requerimiento a implementar/modificar y los equipos de desarrollo se organizan para realizarlo y probarlo, valiéndose de Plastic SCM para guardar su progreso y promocionarlo cada cierto tiempo a IC. Se utiliza también Polarion para la gestión del ciclo de vida de la aplicación, permitiendo la subida de requisitos de usuario, funcionales y no funcionales. Con este software se puede realizar la trazabilidad hasta el código.
2. Se utiliza la herramienta Ágora para la instalación del código y así se permita la realización de las pruebas de integración continua para validar que todos los desarrollos funcionan entre sí. También se utiliza para el versionado de forma automática. Esta subida se realiza cada 2 o 3 días.
3. Se realiza la promoción a Integración con la misma herramienta. En este punto, además de las pruebas de integración se realizan pruebas de regresión, para comprobar que el código anterior funciona de manera correcta con el nuevo añadido. Esta subida se realiza de manera semanal.

4. Se comprueba, por parte del equipo de calidad, todas las pruebas funcionales y de regresión en gran parte de manera automática. Además, se realizan métricas de manera manual de la calidad del software.
5. Por último, el producto pasa al entorno de Producción, que corresponde a cada país, dónde estos tienen su propio ciclo de vida o utilizan directamente esta versión (Core).

1.2 Motivación

La motivación del desarrollo de la herramienta en la que se centra esta memoria aparece en el momento en el que los analistas encargados de subir el desarrollo de IC a Integración se dan cuenta de que el proceso de entrega, por parte de los desarrolladores, de los changesets (una serie de cambios o uno solo) implicados de cada repositorio es propenso a generar errores. Del mismo modo surgió la idea de que se necesitaba un repositorio que albergara información fiable sobre el estado del proyecto en todos los posibles entornos.

Para este proceso se creó un Excel en el que cada repositorio ocupaba una fila y las columnas eran las iteraciones de una release (un conjunto de evolutivos en el progreso de evolución del software), la intersección entre una iteración y un repositorio se rellenaba con el changeset concreto. Como es de suponer, esto genera muchas veces errores a la hora de escribir o leer el changeset, además de todos los inconvenientes ligados a tratar esta información en un Excel y no en una base de datos como tal.

Otra de las motivaciones para la mejora de procesos anteriores era el de la detección de errores de sintaxis en el código a promocioar a IC. Este proceso seguía los siguientes pasos: primero los desarrolladores accedían a la herramienta Plastic SCM y dentro de esta a cada uno de los repositorios implicados en la subida. Aquí se deberían validar la sintaxis y codificación de numerosos archivos PL, lo cual era una inversión de tiempo considerable. Esto tenía que realizarse ya que se detectó un problema en las instalaciones de código mal implementado que no atendían a la normativa aceptada por las herramientas de instalación (construida alrededor de la unidad SQL*Plus de Oracle).

A la hora de realizar las métricas de calidad, el analista debía entrar en diferentes portales de gestión de la herramienta para descargar distintos archivos Excel, realizar algunas consultas en base de datos con un select predefinido y exportar la información a otros archivos Excel, para que finalmente todos estos fueran analizados por una herramienta que muestra gráficas de evolución y calidad. En esta tarea se invertían aproximadamente 4 horas entre búsqueda de información y generación de documentos. Estas extracciones se realizan mediante dos herramientas en constante utilización que son Polarion, para la gestión de las dependencias de las funcionalidades y los requisitos de cada una de estas y Clarity, un repositorio con información relativa a las demandas, incluyendo su nombre, código, dificultad, tiempo de desarrollo, etc.

Un proceso que se repetía de forma constante era el de comprobar que versión de cada repositorio o proyecto estaba alojada en cada entorno. Este era bastante sencillo, pero poco práctico cuando había que comprobar la versión de dos proyectos o más ya que consistía en hacer una consulta en la base de datos concreta y para obtener la línea base

(versión de cada uno de los proyectos en una iteración/release concreta) actual en un entorno concreto, habría que tener guardada una consulta previamente definida o entrar en CMDB, una base de datos de conocimiento poco intuitiva y muy lenta, por cada repositorio en el caso de Preproducción.

Cuando una demanda o petición necesita de una implementación en distintos repositorios, el nombre de la rama en cada uno de estos contiene el de esta demanda. Para conocer que repositorios están involucrados en dicha demanda había que buscar en cada repositorio utilizando la interfaz gráfica, el nombre de esta demanda, lo cual era prácticamente inviable por el tiempo necesario para realizar la búsqueda. Este proceso era utilizado para saber si una demanda había entrado o no en una entrega y detectar y chequear de una manera rápida sus componentes en caso de problemas.

1.3 Solución propuesta

Partiendo de toda la problemática focalizada mayoritariamente en los tiempos de ejecución de distintos procesos manuales y distintos sitios a consultar, se propone centralizarlos en un único portal web que realice todos estos procesos de forma interna y de la manera más automática y eficiente posible.

El portal servirá para ahorrar una cantidad de tiempo considerable en la realización de estas tareas repetitiva, ya que se mostrará información que se actualice automáticamente mediante procesos batch y el usuario no tendrá que buscarla en BD constantemente. También contará con una pasarela en forma de interfaz gráfica para interactuar con la herramienta SCM así como una API REST para poder ser utilizada por otros programas y así ahorrar tiempo y reducir el riesgo de errores. Es decir, una herramienta que dé solución a cada uno de los problemas planteados en el apartado anterior.

Se realizará un portal web en PHP, utilizando Ajax para dar dinamismo, que llamará a procesos batch internos para la actualización de documentación en horarios fuera de oficina. Se lanzará un analizador sintáctico (que también se deberá desarrollar) de manera instantánea, para corregir o detectar errores en el código PL/SQL, así como las descargas del SCM llamando al comando. Por último tendrá un repositorio con cierta información requerida de manera que quede centralizada y con fácil acceso (direcciones de servidores y si están disponibles o no, log de los entornos, la línea base de IC, Integración y Preproducción,

Este portal estaría alojado en un servidor Apache instalado en Windows. Como gestor de bases de datos se utilizará MySql. A este servidor solo podrá acceder el desarrollador y a la dirección del portal únicamente se podrá acceder desde la red interna de la empresa, asegurando así que cualquiera que esté conectado pueda acceder a la información, siendo indiferente el puesto que ocupe en el departamento. Además, se utilizará el programador de tareas de Windows para lanzar procesos nocturnos (batch) para la actualización de bases de datos interna accediendo a la información requerida, cargando la información y tratando los archivos que necesiten un tiempo de ejecución elevado.

1.4 Metodología utilizada

Para comenzar con este apartado se va a definir la metodología software y a continuación se desarrollará la utilizada para este proyecto.

La metodología de desarrollo del software es la estrategia que marcará el proceso de análisis, diseño, desarrollo y verificación del programa a construir y tendrá que ver con la comunicación entre las partes involucradas, los tiempos de construcción de estas y el intercambio de información [1].

Dada ya una definición de metodología y después de investigar cual sería la idónea para este proyecto se ha tomado la decisión de utilizar el modelo de cascada. Esta metodología consiste en una serie de etapas que se siguen de forma secuencial, partiendo de la toma de requisitos y finalizando en la entrega del software.

Se ha optado por esta metodología ya que es ideal cuando "los requisitos y características de un software pueden ser claramente definidos durante la fase conceptual" [2].

Cada una de las partes del proyecto son completamente independientes y están perfectamente definidas, ya que consiste en reproducir un proceso manual de forma automática y debiendo obtener el mismo resultado, por lo que se adecua a la definición citada anteriormente.

Se ha decidido construir cada parte del proyecto de forma independiente, pero integrándolas en la misma herramienta, por lo que se seguirá la metodología para cada uno de estos módulos y cada evolutivo irá integrado en el portal desarrollado.

La primera de las entregas será la más compleja, ya que se deberá crear toda la infraestructura donde irá la funcionalidad requerida, y las sucesivas se integrarán en esta para mejorarla.

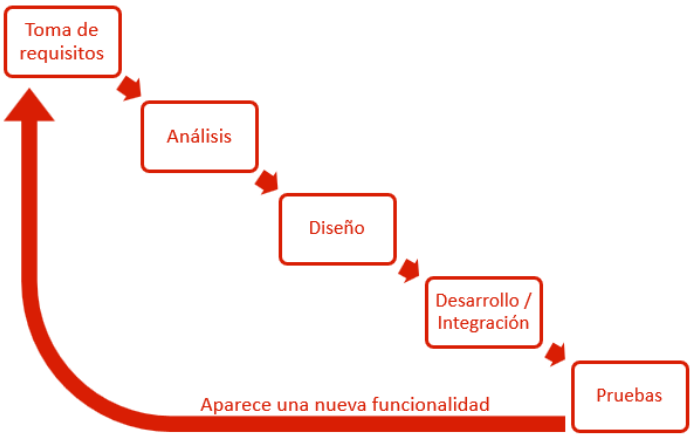


Figura 1.1: Metodología en cascada
Fuente: propia

El proceso que seguirá cada uno de estos módulos será el siguiente [2]:

1. **Toma de requisitos.** El proceso comenzará con una entrevista con una de las personas encargadas en hacer la tarea concreta de forma manual y repetitiva. En esta fase se tomará nota de todas las acciones que debe seguir, el orden y el resultado obtenido en el formato concreto, dejando todo el proceso claro y sin lugar a duda.
2. **Análisis de requisitos.** En este punto se deberá estudiar la viabilidad del requisito ya que ciertas tareas no se pueden automatizar. Si esto fuera así se comunicaría al entrevistado de la primera etapa para que autorice la automatización parcial de la tarea y no su totalidad.
3. **Diseño.** En esta etapa se investigarán las herramientas que se van a necesitar y el modo de utilización. Además, se diseñará la interfaz y la forma de actualizar la información si fuera necesario.
4. **Desarrollo.** Se codifica la automatización y la interfaz a la hora de mostrar la información requerida o la forma de empaquetado y descarga en el caso que corresponda. Durante esta etapa se integrará el módulo en el portal y se comprobará que genera la información requerida de forma correcta.
5. **Uso del portal / pruebas.** La última de las etapas consistirá en una demo con la persona de la primera etapa de tal forma que verifique la generación de la información y aprenda a utilizar la herramienta.

Esta metodología se deberá seguir en el caso de que, conforme el portal vaya avanzando, surjan más procesos que se quieran automatizar.

1.5 Organización de la memoria

Esta memoria se ha organizado de manera que siga las etapas de un proceso de desarrollo del software:

- **Capítulo 1.** Se contextualiza el proyecto, se resumen las problemáticas encontradas antes de su realización, se detalla la metodología seguida y se muestra la estructura del documento.
- **Capítulo 2.** Se explica en detalle algunas de las tecnologías que se utilizan en el departamento donde se realiza el proyecto, así como las metodologías utilizadas en el mismo.
- **Capítulo 3**. Se detallan todos los requisitos que ha de tener la aplicación, comenzando por los requisitos de usuario, los funcionales y los no funcionales.
- **Capítulo 4.** Se habla sobre las tecnologías y herramientas utilizadas en el desarrollo del portal web.
- **Capítulo 5.** Se comenta como han sido las pruebas realizadas, la funcionalidad final del portal y cómo será el mantenimiento y soporte.
- **Capítulo 6.** Se muestra las conclusiones obtenidas después de la finalización del proyecto y las posibles nuevas líneas de trabajo tras la instalación de este.

2 ESTADO DEL ARTE

En este capítulo se detallarán las tecnologías en uso y las formas de trabajo utilizadas en el departamento donde se creará el software del que trata este documento.

2.1 Tecnologías en uso

Se destacan cuatro tecnologías que aparecen constantemente y de las que depende esta herramienta que se va a crear.

2.1.1 Plastic SCM

Cuando más de una persona (incluso cuando es únicamente una) trabaja sobre un mismo proyecto necesita una herramienta que gestione de forma automática todos los cambios, así como encargarse de optimizarlos y organizarlos ya sea de manera temporal, por equipos de desarrollo o por líneas de trabajo [3].

Siempre es importante tener una copia de seguridad (o backup) del proyecto que se está creando, así como un "sitio" donde se guarde este software y pueda descargarse ya sea de forma total como parcial, para instalarlo, hacer pruebas con él o simplemente para seguir desarrollándolo. Para realizar esta tarea existen múltiples herramientas como Git, SVN o Plastic SCM.

Plastic SCM es la herramienta de control de versiones que se viene utilizando en este caso. Al ser un sistema distribuido, permite que los equipos de desarrollo trabajen en un mismo proyecto de manera ordenada y segura desde distintas localizaciones, incluso sin conexión, por lo que el proyecto estará disponible y accesible en todo momento, dando la posibilidad de crear un servidor replicado en el equipo personal vinculado al principal

[4]. De esta manera se mantiene la filosofía de tener siempre el propio desarrollo sincronizado con el de los demás, evitando así conflictos a la hora de promocionar el software a los entornos

Una de las características más notables de la herramienta es su interfaz gráfica. Como se muestra en la imagen se puede observar la claridad en la organización del proyecto, con ramas que corresponden a líneas de desarrollo distintas [5], de las cuales salen subramas para desarrollar problemas concretos.

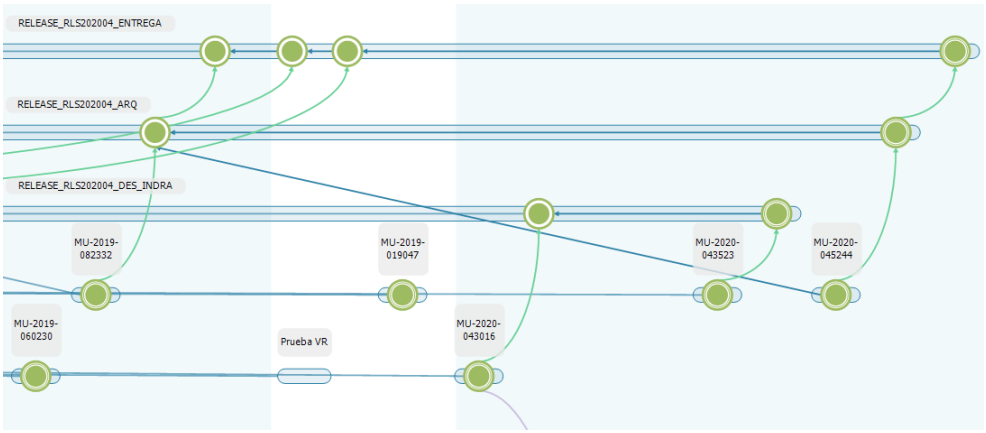


Figura 2.1: Captura de interfaz Plastic SCM
Fuente: propia

El concepto de rama es uno de los más importantes a la hora de hablar de este controlador de versiones. Una de las particularidades es que cada una guarda únicamente los ficheros o directorios que se han modificado desde la rama padre ya que en su creación se generan objetos vacíos en lugar de una copia de los datos. De esta manera se pueden crear muchas más ramas en un mismo repositorio sin que se vea decrementado el rendimiento [5].

El flujo de trabajo en cada release quedaría de la siguiente forma:

1. Se crea una rama release que hereda la información de la rama integration (rama principal).
2. Cada equipo de desarrollo crea una rama que hereda de la rama release para sus propias modificaciones.
3. Cada equipo de desarrollo puede crear tantas ramas como necesite para gestionar el software según tengan organizada su metodología de trabajo.
4. Cada cierto tiempo los desarrollos se subirán a la rama principal de cada equipo.
5. Una vez a la semana se subirán los desarrollos a la rama release.
6. Cuando una release termina, se sube todo el desarrollo de rama a la de integration.

Por normativa se ha de nombrar la rama concreta donde se está desarrollando con el nombre de la petición o requerimiento a implementar, aunque de esta rama “requisito” puedan surgir más.

Otro concepto muy importante que atañe a este proyecto es el de “changeset” (o “cs” en su forma diminutiva): una serie de cambios, o uno solo, en una rama concreta. Como hemos definido antes, solo se almacenan los cambios, siendo el resto de los ficheros objetos vacíos indexados al padre o sucesivos (en el caso de no haber sido modificado).

Es importante mencionar que cada changeset tiene un identificador en forma de entero que empieza desde el 1 y que se incrementa de forma secuencial, sin tener que ver con la rama en la que se encuentre y un label que recibe el nombre de la versión, en nuestro caso solo se versiona cuando hay una promoción al siguiente entorno. Derivando de este concepto se ha de explicar el de "diferencias" entre changesets. Como se puede suponer, serán de tres tipos: Archivos/Directorios Modificados (M), eliminados (D) o creados (C). De esta manera, por línea de comandos o gracias a la interfaz gráfica, se puede llevar un control de estos tres tipos de diferencias entre dos changeset cualquiera de un mismo repositorio.

Tanto los conceptos de diferencias como los de identificador y versión serán utilizados en una pantalla que hará de pasarela para interactuar con Plastic sin tener que saber de comandos y sin abrir la herramienta en sí.

Sobre la línea de comandos, Plastic ofrece un comando (cm) muy potente que se instala a la vez que la herramienta gráfica e incluye una serie de opciones, de las cuales las más utilizadas en este proyecto serán cm diff y cm find para consultar las diferencias entre changesets y buscar las ramas que incluyan el nombre de una demanda concreta [6]. Gracias a este comando se pueden realizar de forma automática las mismas tareas que se harían desde la GUI, lo que será de mucha utilidad a lo largo del desarrollo de la herramienta.

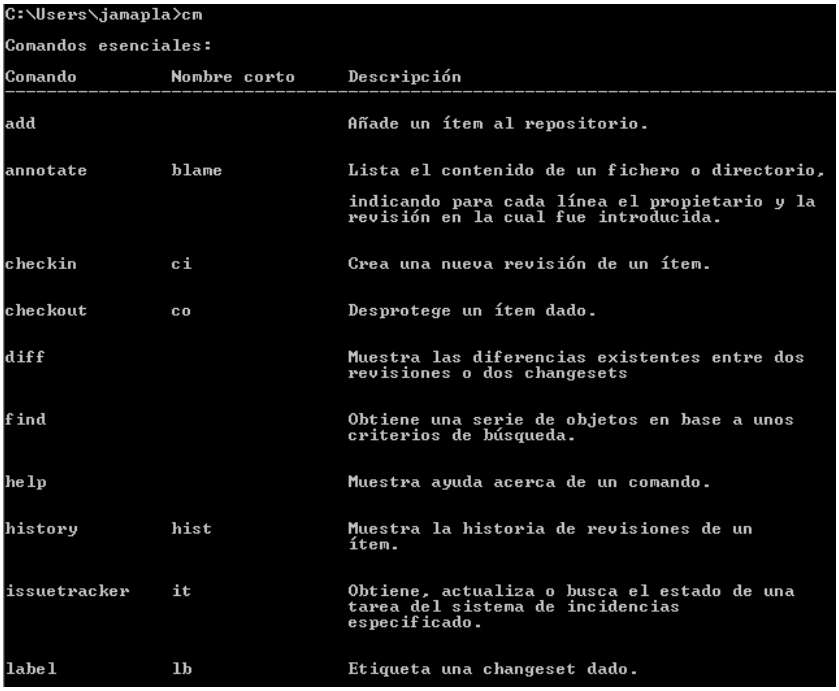


Figura 2.2: Captura de CMD (cm)
Fuente: propia

2.1.2 PL/SQL

El aplicativo de gestión de activos de empresa que se desarrolla y mantiene, tiene en su back-end una base de datos escrita en PL/SQL que significa “extensiones de lenguaje de procedimiento SQL” [7]; es decir, toda la lógica de negocio y de datos está programada en este lenguaje. Mantiene las mismas sentencias que SQL pero con la diferencia de que este ejecuta una sola consulta a la vez y PL/SQL puede ejecutar un bloque entero de código [8].

Es un lenguaje de programación perteneciente a Oracle por lo que solo se puede usar en bases de datos relacionales de esta compañía [7]. Existe una base de datos Oracle para cada entorno, excepto desarrollo que necesita varias, dedicadas cada una de ellas la realización de distintas pruebas.

Se cuenta con una herramienta encargada de las promociones llamada Ágora que, dados los changeset y repositorios de Plastic, desempaqueta el proyecto, hace las pruebas de integración, etiqueta y despliega en el siguiente entorno. Gracias a este proceso se almacenan en una tabla en base de datos las versiones promocionadas, con lo que podemos consultar fácilmente cuál es la que está desplegada en un entorno concreto de un repositorio en particular. Incluso se podrían hacer consultas que mostraran las últimas versiones desplegadas en cada uno de estos entornos.

También en base de datos se almacenan registros relacionados con la productividad y la calidad del aplicativo. Esto nos serviría para poder exportar los datos a un Excel gracias al IDE (entorno de desarrollo integrado) [9] que ofrece Oracle SQL Developer y analizar estos gráficos de una forma más sencilla.

2.1.3 Herramienta de instalación y SQL*Plus

Los ficheros PL/SQL se instalan en base de datos mediante una herramienta interna que los descarga, los ordena y finalmente llama a SQL*Plus para instalarlos.

SQL*Plus es una herramienta de Oracle que viene incluida con cada instalación de su base de datos, ofrece una interfaz de línea de comandos y una gráfica [10], en este caso se utiliza la interfaz de línea de comandos.

Existe una normativa de requisitos que deben contemplar todos los ficheros tanto SQL como PL/SQL para poder instalar con la herramienta. En este apartado se destacan los aspectos para tener en cuenta a la hora de utilizar este instalador. Hay algunas sentencias que al ser utilizadas por línea de comandos pueden ocasionar timeout o errores de ejecución.

Uno de los errores más frecuentes es el de añadir CONNECT, que básicamente conecta un usuario a una base de datos de Oracle y ejecuta una serie de perfiles predefinidos [11], al inicio de los ficheros a instalar. Los desarrolladores lo incluyen ya que trabajan con la interfaz gráfica en la que al iniciar sesión se han de conectar a la base de datos y al ejecutarlo no da ningún problema. Pero al utilizar la línea de comandos,

SQL*Plus se queda esperando a que se introduzcan las credenciales, lo que genera un timeout en la instalación.

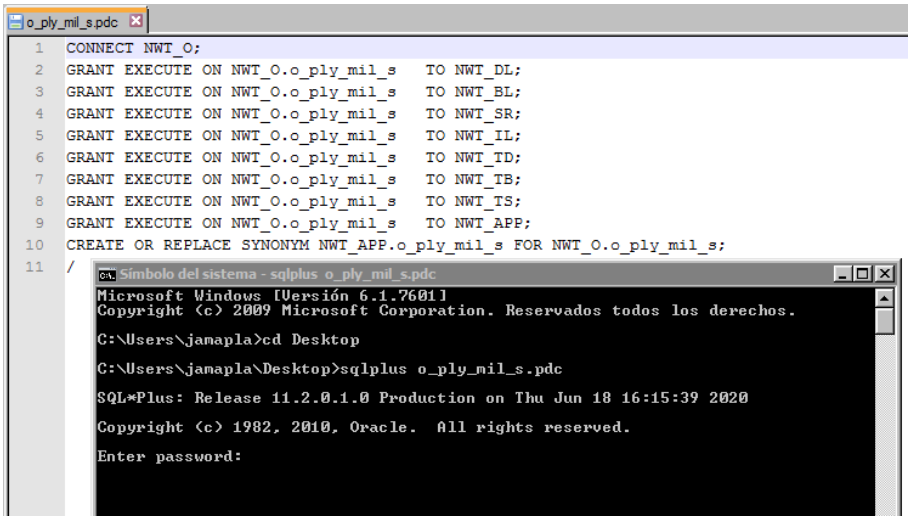


Figura 2.3: Captura error CONNECT
Fuente: propia

También ocurre un timeout cuando en un archivo se añade un ampersand (&), aunque aparezca en un comentario del script, ya que el instalador queda esperando a que se introduzca un valor que sustituya la cadena que lo acompaña [12].

Otro de los errores frecuentes es el uso de EXIT. Al utilizar, como hemos dicho antes, la interfaz gráfica y estar tratando un único archivo a la vez, normalmente los desarrolladores dan por válido un archivo terminado con EXIT, ya que no sale del programa. El problema viene cuando este fichero se ejecuta entre otros dos o más archivos. La sentencia EXIT termina el proceso SQL*Plus y devuelve el control al sistema operativo [13], lo que origina que, tras la instalación de ese paquete concreto, termine el proceso de instalación.

Cuando un script termina con "/", éste ejecuta todo el bloque PL/SQL almacenado en el buffer, por lo tanto funciona como un comando RUN [14]. La manera correcta de utilizar la barra es escribir "/" en una línea sola después de cada bloque PL/SQL [15]. De esta manera se pueden concatenar archivos en uno solo y que el instalador los ejecute por bloques (esto es lo que hace la herramienta previa a SQL*Plus).

Cuando se desarrolló esta herramienta no se tuvo en cuenta la necesidad de introducir la barra y cuando un script no la lleva, el instalador da un error y no se puede ejecutar correctamente.

Por último, la herramienta previa a SQL*Plus, necesita que los archivos de PL/SQL estén codificados en UTF-8 (8-bit Unicode Transformation Format) [16], esto es debido a que la empresa trabaja con multilenguaje y este tipo de codificación proporciona una manera consistente y estandarizada de intercambio y codificación de texto de forma internacional [17]. La particularidad de este tipo de codificación es la utilización de una longitud variable (de 1 a 4 bytes) de los caracteres y la compatibilidad con la codificación estándar ASCII (American Estándar Code for Information Interchange) [17] [18], cuya longitud es delimitada por únicamente 7 bits más uno de paridad [18].

Todos estos detalles tendrán un peso muy importante en una de las herramientas a desarrollar en este proyecto.

2.1.4 Sharepoint

Ya hemos hablado de la forma de compartir la evolución del código mediante Plastic SCM, pero cuando lo que se quiere compartir no es código como tal sino documentos, accesos directos o scripts, se utiliza Sharepoint.

Sharepoint es una herramienta de Microsoft la cual está orientada a compartir y administrar contenido, conocimientos y aplicaciones de manera que se pueda trabajar en equipo con más facilidad y rápidamente, de tal forma que cualquier miembro de la organización pueda tener acceso [19].

Actualmente cada equipo de trabajo gestiona su Sharepoint de la forma que cree conveniente: algunos incluyen gráficos y tienen un diseño más elaborado, otros suben documentos interesantes y de utilidad de una manera más sencilla e incluso hay algunos que prefieren otros métodos de compartición de documentos.

Hay dos casos importantes en los que vamos a trabajar en este proyecto: el registro de instalaciones del entorno de Integración Continua (IC) y el Sharepoint del equipo de mantenimiento de las peticiones de implementación.

El registro de instalaciones se almacena en un archivo Excel en el que las filas corresponden a cada uno de los repositorios del aplicativo y las columnas corresponden a las iteraciones de una release (hojas del documento) concreta. La intersección entre fila y columna se rellena de forma manual con el changeset que se ha promocionado a IC. Además se introducen las peticiones implementadas en la columna (iteración) correspondiente. Este Excel se sube de forma manual al sharepoint para que cualquiera pueda descargarlo o consultarlo online y averiguar qué changeset es el correspondiente a una iteración/repositorio concreto. Esta subida se hace aproximadamente dos veces a la semana.

Cuando nace un nuevo requisito a implementar se crea una petición y se almacena con un identificador, el nombre del encargado, una descripción, etc. Y se modifica el estado de la petición (abierta, cerrada, terminada). Estas peticiones se suben al Sharepoint en formato json, un formato de intercambio de datos caracterizado por una colección de pares nombre/valor donde que es de simple interpretación por programas [20]. También gestionan toda esta información en determinados archivos Excel.

Para finalizar, mencionar que una de las características más interesantes que se utilizarán para la implementación del portal es que Sharepoint se puede vincular como unidad de red de una forma muy sencilla [21], lo que permite que desde cualquier programa se pueda acceder a los documentos subidos de forma pública así como actualizarlos o subir nuevos documentos de forma automática como si se tratase de una unidad de almacenamiento más del servidor.

2.2 Entornos

Al tratarse del desarrollo y mantenimiento de un software de gestión, es muy importante la elección de los entornos [22], debemos saber cuáles son los tipos y para que se utilizan cada uno de ellos.

Se define entorno como la infraestructura orientada a las tareas requeridas por el software dependiendo del escenario en el que se encuentren [23], ya sea desarrollo, pruebas, etc. Conforme el software avanza de entorno a entorno el nivel de estabilidad del producto se afianza [23].

Se diferencian cinco entornos: Desarrollo, Integración Continua, Integración, Preproducción y Producción.

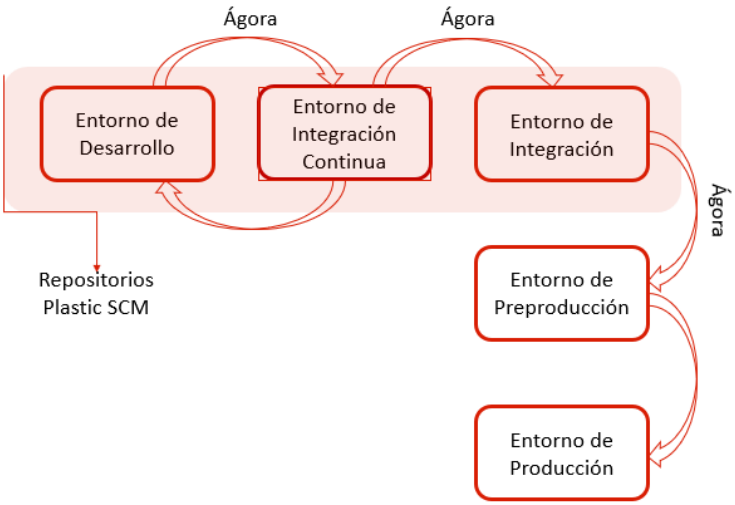


Figura 2.4: Promoción entre entornos
Fuente: propia

Antes de comenzar es necesario definir el concepto de demanda o petición: una funcionalidad nueva, una corrección o un cambio a desarrollar cuyo identificador lleva siempre la misma nomenclatura. Puede ser para un país concreto, un módulo, un objeto o un atributo para un departamento; así mismo, es la unidad mínima de software entregable.

Previo al entorno de desarrollo se realiza una planificación de cuáles son las demandas que se implementarán en una release, de cómo se dividirá en las distintas iteraciones y qué peticiones irán en cada una de éstas. A continuación se define un calendario de entregas de las demandas a los entornos y cada momento de entrega marca una iteración nueva.

Una vez realizada la planificación se procede al alta de las demandas en la herramienta Polarion, donde se almacena el diseño técnico de cada requisito y sus dependencias. Seguidamente se crea la rama release y las de desarrollo en Plastic.

Cuando este escenario está creado, el equipo de desarrollo recibe las demandas y comienza su trabajo.

Desarrollo

El proceso de desarrollo comienza con la organización interna de cada equipo sobre quién realizará cada parte de las demandas, para ello se deberá acceder a Polarion para interpretar el análisis de la funcionalidad y construirlo. A continuación, los desarrolladores crearán las ramas Plastic necesarias para implementar cada funcionalidad, que heredarán de la rama del equipo concreto. Es importante que en el nombre de la rama creada incorpore el de la demanda involucrada para facilitar la organización y el seguimiento de éstas.

Una vez creado el entorno de trabajo, los desarrolladores pueden empezar a codificar la/s demandas asignadas utilizando el IDE que se ajuste más a cada parte del aplicativo. De forma habitual se utiliza Sublime Text para los desarrollos en Javascript, Eclipse para la parte de Java y PL/SQL Developer para el back-end.

Una vez realizado la implementación que se le asignó a cada desarrollador, deberán realizar las pruebas unitarias del requisito, así como las que se consideren necesarias para corroborar que funciona correctamente.

Una vez se han completado todos los pasos anteriores y cuando llega la fecha de entrega según el calendario, los desarrolladores subirán sus implementaciones a la rama del equipo y de ahí las demandas contempladas en esa iteración promocionan a la de Integración Continua.

Integración Continua

Cuando cada equipo de desarrollo ha promocionado las modificaciones de cada implementación en cada uno de los repositorios correspondientes, se lleva a cabo la subida a Integración Continua (IC), donde se realizan una serie de pruebas de integración y el versionado con la siguiente nomenclatura: XX.YY.ZZ.PP, donde XX corresponde a los dos últimos dígitos del año, YY a la release, ZZ a la iteración y PP en el caso de ser un parche.

Este entorno se utiliza para comenzar de forma gradual la migración de la metodología que se venía utilizando hacia DevOps, que consiste en la coordinación y colaboración de manera integrada entre los roles de desarrollo (Development) y los de calidad y seguridad y pruebas (Operaciones u operations) [24] [25].

Dentro de la metodología DevOps encontramos el concepto de CI/CD (Continuous Integration / Continuous Delivery-Deployment). Se utiliza para implementar, integrar y distribuir las funcionalidades de cada equipo de trabajo con mucha más frecuencia de manera automática [26].

La herramienta utilizada es Ágora, mencionada anteriormente, basada en Jenkins: una de las aplicaciones de código abierto más utilizadas para la gestión de la metodología DevOps, que permite la integración de distintos plugins para especializar mejorar el ciclo de desarrollo [27].

Una vez hecha la integración a IC, el encargado de la subida rellenará el Excel de registro de instalaciones y se subirá a Sharepoint. Con esta información y la subida realizada, se procede a las pruebas de regresión para comprobar que, habiendo verificado

el programa e introducidas nuevas modificaciones, no falta ninguna funcionalidad y no existen defectos [28]. Si el software pasa los test, la rama promociona a integración.

Integración

Se realiza la subida al entorno de Integración cuando sea la fecha determinada por el calendario de planificación. La herramienta Ágora será la encargada de realizar la promoción. En este punto se realizan pruebas de integración, para comprobar que el código funciona añadiendo las modificaciones, y de regresión para comprobar que los nuevos requisitos instalados no han dañado a los que ya estaban.

Los analistas y el departamento de Calidad empiezan a realizar test y métricas concretas, así como pruebas funcionales. Si el sistema llega a este momento satisfactoriamente, se promociona al entorno de Preproducción (PRE).

Preproducción

Se realizarán el resto de las pruebas por parte del equipo de Calidad [23], utilizando una serie de métricas y test predefinidos gracias a LoadRunner. Cuando finalicen, deberán autorizar el paso al entorno de producción.

Muchas de las métricas realizadas, en cuanto a la calidad del software, se realizan de forma manual cuando idóneamente deberían no serlo. Procesos como descargas de documentos de Polarion, de Sharepoint o consultas a base de datos se realizan semanalmente para analizar el buen funcionamiento de la metodología, el cumplimiento del calendario marcado y sobre todo el óptimo funcionamiento del software. Gracias al portal web a desarrollar se ahorrará mucho tiempo convirtiendo estas tareas en automatismos.

Producción

Este último entorno corresponde a cada país en el que se instala la solución. Cada uno de estos tiene su propio ciclo de vida del aplicativo, pueden utilizar el Core, la versión que se promociona sin necesitar ninguna modificación (como es el caso de España o Malta) o bien lo pueden personalizar, de manera que ellos gestionan sus las promociones entre sus propios entornos de la forma que tengan estipulada hasta llegar a Real.

3 ANÁLISIS DE REQUISITOS

En este capítulo se recopilará el detalle de todos los requisitos de los que ha de constar el proyecto a desarrollar. Primeramente se presentarán los requisitos de usuario, de los cuales se partirá para la documentación de los funcionales y no funcionales.

3.1 Requisitos de usuario

A continuación se mostrará el listado correspondiente a los requisitos de alto nivel obtenidos tras las conversaciones con el cliente, en este caso un representante de cada tarea que se ha de automatizar.

- **RU1**. El sistema permitirá consultar la línea base de los entornos de IC, Integración y PRE de manera manual.
- **RU2**. Se ofrecerán servicios REST documentados que permitan consultar la línea base de un entorno (IC, Integración y PRE) o de una release
- **RU3**. El usuario deberá poder gestionar (lo que supone poder realizar registros, modificaciones y consultas) las instalaciones de forma manual.
- **RU4**. Se ofrecerán servicios REST documentados que permitan gestionar (lo que supone poder realizar registros, modificaciones y consultas) las instalaciones.
- **RU5**. Se deberá mostrar las direcciones de los servidores de cada entorno así como su disponibilidad.
- **RU6**. El sistema permitirá corregir ficheros PL/SQL para ajustarlos al formato predefinido.
- **RU7**. Se deberá poder consular los proyectos en los que se está desarrollando una demanda (unidad mínima de software entregable).
- **RU8**. Se ofrecerán servicios REST documentados para averiguar la extensión temporal de una release y los changeset involucrados en una instalación.

- **RU9**. Se generarán los documentos implicados en las métricas de Calidad.
- **RU10**. Se ofrecerán los logs de las aplicaciones en los distintos entornos.

3.2 Requisitos funcionales

Los requisitos funcionales describen los servicios que debe proporcionar el sistema [29]. Para definirlos se ha tenido en cuenta las entradas al servicio, el flujo que sigue el proceso y las salidas esperadas de cara a la facilidad de implementación. Se enlaza además el requisito de usuario del que a partido cada requisito funcional.

RF1: Mostrar línea base IC

Código Requisito Funcional: RF1.
Nombre: Mostrar línea base IC.
Requisito de Usuario de partida: RU1.
Descripción: Muestra el último changeset correspondiente al entorno de Integración Continua de cada repositorio Plastic.
Entradas: Ninguna.
Flujo:

1) El sistema se conecta a la base de datos de IC.
2) Se ejecuta la consulta de todos los changeset de cada repositorio en orden.
3) Se filtran los resultados y se obtiene el último changeset de cada repositorio.
4) Se almacenan los resultados en la base de datos del sistema.
5) El frontal del sistema muestra los resultados de la base de datos interna.

Salida: Presenta una tabla con dos columnas donde en la primera columna aparezca el nombre de los repositorios y en la segunda columna el último changeset correspondiente al entorno de Integración Continua.

RF2: Mostrar línea base Integración

Código Requisito Funcional: RF2.
Nombre: Mostrar línea base Integración.
Requisito de Usuario de partida: RU1.
Descripción: Muestra el último changeset correspondiente al entorno de Integración de cada repositorio Plastic.
Entradas: Ninguna.
Flujo:

1) El sistema se conecta a la base de datos de Integración.
2) Se ejecuta la consulta de todas las versiones de cada repositorio en orden.
3) Se filtran los resultados y se obtiene la última versión de cada repositorio.
4) Se traducen las versiones a changesets de Plastic.
5) Se almacenan los resultados en la base de datos del sistema.
6) El frontal del sistema muestra los resultados de la base de datos interna.

Salida: Presenta una tabla con una columna al lado de la columna de IC correspondiente a RF1 en la cual aparezca el último changeset correspondiente al entorno de Integración de cada repositorio Plastic.

RF3: Mostrar línea base Preproducción

Código Requisito Funcional: RF3.
Nombre: Mostrar línea base Preproducción
Requisito de Usuario de partida: RU1.
Descripción: Muestra la versión correspondiente al entorno de Preproducción de cada proyecto.
Entradas: Ninguna.
Flujo:

1) Se accede a cada dirección de cada proyecto en el servidor de Preproducción.
2) Se analiza el código del frontal para encontrar la versión instalada de cada proyecto.
3) Se almacenan los resultados en la base de datos del sistema.
4) El frontal del sistema muestra los resultados de la base de datos interna.

Salida: Presenta una tabla con una columna al lado de la columna de Integración correspondiente a RF2 en la cual aparezca la versión correspondiente al entorno de Preproducción de cada proyecto.

RF4: Registro de instalación

Código Requisito Funcional: RF4.
Nombre: Registro de instalación.
Requisito de Usuario de partida: RU3.
Descripción: Permite a un integrador registrar cada uno de los changeset promocionados a un entorno concreto.
Entradas:

1) Entorno
2) Release
3) Iteración
4) Demandas involucradas
5) Descripción (opcional)
6) Changeset de cada repositorio (mínimo el changeset de un repositorio)

Flujo:

1) Acceder al Sharepoint de mantenimiento de peticiones.
2) Mostrar solo las demandas que estén abiertas.
3) Comprobar que no existe la iteración en ese entorno y release.
4) Registrar en base de datos interna todas las entradas.
5) Acceder al Excel de IC (en el caso de que el entorno sea IC), crear una nueva columna (u hoja si es una nueva release) y rellenar los datos de la entrada.
6) Guardar el Excel en el sharepoint y mostrar la salida en el formato que corresponda.

Salida: Muestra un mensaje de Error en caso de haber habido algún inconveniente o falta de dato o un mensaje resumen de la instalación registrada y un enlace al Excel actualizado para su descarga en caso de no haber ningún fallo.

RF5: Modificar instalación

Código Requisito Funcional: RF5.
Nombre: Modificar instalación.
Requisito de Usuario de partida: RU3.
Descripción: Permite a un integrador modificar changeset, demandas o descripción de una instalación previamente creada.

Entradas:

1) Entorno
2) Release
3) Iteración
4) Demandas involucradas (opcional)
5) Descripción (opcional)
6) Changesets (opcional)

Flujo:

1) Mostrar desplegable de entornos en base de datos y elegir uno.
2) Mostrar desplegable de releases y elegir una.
3) Mostrar desplegable de iteraciones y elegir una.
4) Registrar en base de datos interna todas las entradas.
5) Acceder al Excel de IC (en el caso de que el entorno sea IC) y modificar la instalación deseada.
6) Guardar el Excel en el sharepoint y mostrar la salida en el formato que corresponda.

Salida: Muestra un mensaje de Error en caso de haber habido algún inconveniente o falta de dato o un mensaje resumen de la instalación modificada y un enlace al Excel modificado para su descarga en caso de no haber ningún fallo.

RF6: Consultar una instalación

Código Requisito Funcional: RF6.
Nombre: Consultar una instalación.
Requisito de Usuario de partida: RU3.
Descripción: Permite poder ver los changeset involucrados en cada repositorio de una instalación, los últimos changeset almacenados en un entorno o en una release previamente registrada.
Entradas:

1) Entorno
2) Release (opcional)
3) Iteración (opcional)

Flujo:

1) Mostrar desplegable de entornos en base de datos y elegir uno.
2) Mostrar desplegable de releases y elegir una. (opcional).
3) Mostrar desplegable de iteraciones y elegir una. (opcional).
4) Al pulsar el botón de “Consultar” hará la query 1, 2 o 3 dependiendo de si se consulta por entorno, release o iteración.
5) Muestra la información obtenida de base de datos.

Salida: Muestra los últimos changeset promocionados a un entorno concreto si no se especifica nada más. Muestra los últimos changeset promocionados a un entorno concreto referidos a una release concreta si se especifica. Muestra una instalación concreta si se especifican las tres entradas.

RF7: Mostrar proyectos involucrados de una demanda

Código Requisito Funcional: RF7.
Nombre: Mostrar proyectos involucrados de una demanda.
Requisito de Usuario de partida: RU7.

Descripción: Muestra todas las ramas de los repositorios de Plastic en los que se está desarrollando una demanda concreta.
Entradas: Nombre de la demanda.
Flujo:

1) Lee el nombre de la demanda.
2) Llama al comando cm find de Plastic en cmd.
3) Guarda el resultado, lo formatea y lo imprime en pantalla.

Salida: Muestra el nombre de cada una de las ramas Plastic correspondientes a los repositorios donde aparece la demanda.

RF8: Mostrar disponibilidad de los servidores de cada entorno

Código Requisito Funcional: RF8.
Nombre: Mostrar disponibilidad de los servidores de cada entorno.
Requisito de Usuario de partida: RU5.
Descripción: Comprueba la disponibilidad de los servidores de cada entorno y muestra de forma ordenada las direcciones asociadas informando de su estado.
Entradas: Ninguna.
Flujo:

1) Get a cada servidor de cada entorno.
2) Analizar el código de respuesta del servidor.
3) Mostrar un led verde si el código es 200 y un led rojo si no lo es.

Salida: Muestra un led verde o rojo al lado de cada dirección de los servidores de cada entorno.

RF9: Descargar diferencias de Plastic

Código Requisito Funcional: RF9.
Nombre: Descargar Diferencias de Plastic.
Requisito de Usuario de partida: RU6.
Descripción: Permite descargar el contenido de un changeset (o las diferencias entre dos) y comprobar/corregir los archivos de PL/SQL para que se adapten al formato adecuado.
Entradas:

1) Changeset o versión a descargar
2) Changeset o versión para descargar diferencias (opcional)
3) Modo comprobación de archivos (opcional)
4) Modo corrección de archivos (opcional)

Flujo:

1) Seleccionar descargar por changeset o por versión.
2) Seleccionar si se desea o no comprobar o corregir el formato de los archivos PL/SQL.
3) Guardar los changeset/versiones.
4) Llama al comando cm diff de Plastic en cmd
5) Guarda las diferencias y parsea (o no) en modo análisis (o corrección) cada uno de los archivos PL/SQL.
6) Empaqueta los archivos y un log y muestra un enlace de descarga del zip, los archivos y el log del parser.

Salida: Muestra un mensaje de Error en caso de haber habido algún inconveniente o falta de dato o un enlace al zip que contiene los archivos deseados y un log en el caso de querer comprobar/corregir el formato de los archivos.

RF10: Generación de documentación para Calidad

Código Requisito Funcional: R10.
Nombre: Generación de documentación para Calidad.
Requisito de Usuario de partida: RU9.
Descripción: Muestra un enlace de descarga de un zip con la documentación necesaria para ejecutar el análisis de calidad del software.
Entradas: Ninguna
Flujo:

1) Acceder a la base de datos de IC.
2) Ejecutar consulta predefinida y guardar los datos obtenidos en un Excel con el formato adecuado.
3) Ejecutar consulta predefinida en SQL*Plus y guardar los datos obtenidos en un Excel con el formato adecuado.
4) Acceder al Sharepoint de mantenimiento de peticiones.
5) Parsear los Excel, cribar la información y guardar los datos en un Excel con el formato adecuado.
6) Acceder al Sharepoint de IC y descargar la última versión del Excel con el registro de Instalaciones.
7) Empaquetar todos los ficheros Excel en un zip y mostrar un enlace de descarga.

Salida: Un enlace a la descarga del zip que contiene la información deseada.

RF11: Servicio REST GET resumen temporal release

Código Requisito Funcional: RF11.
Nombre: Servicio REST GET resumen temporal release.
Requisito de Usuario de partida: RU8.
Descripción: Ofrece un servicio REST para obtener el número de iteraciones de una release, la fecha de inicio y finalización de esta. Solo para IC.
Entradas:

1) Release
2) date="true"

Flujo:

1) Consulta en base de datos interna con la release.
2) Convierte los datos de la consulta en JSON y lo devuelve.

Salida: Archivo JSON con la información solicitada.

RF12: Servicio REST GET línea base de un entorno

Código Requisito Funcional: RF12.
Nombre: Servicio REST GET línea base de un entorno.
Requisito de Usuario de partida: RU2.
Descripción: Ofrece un servicio REST para obtener la línea base (últimos changeset registrados de cada repositorio) de un entorno dado.
Entradas:

1) Entorno
2) lb="true"

Flujo:

1) Consulta en base de datos interna con el entorno.
2) Convierte los datos de la consulta en JSON y lo devuelve.

Salida: Archivo JSON con la información solicitada.

RF13: Servicio REST GET línea base de una release

Código Requisito Funcional: RF13.
Nombre: Servicio REST GET línea base de una release.
Requisito de Usuario de partida: RU2.
Descripción: Ofrece un servicio REST para obtener la línea base (últimos changeset registrados de cada repositorio) de una release concreta.
Entradas:

1) Entorno
2) Release

Flujo:

1) Consulta en base de datos interna con el entorno y la release.
2) Convierte los datos de la consulta en JSON y lo devuelve.

Salida: Archivo JSON con la información solicitada.

RF14: Servicio REST GET changesets de una instalación

Código Requisito Funcional: RF14.
Nombre: Servicio REST GET changesets de una instalación.
Requisito de Usuario de partida: RU4.
Descripción: Ofrece un servicio REST para obtener la línea base (últimos changeset registrados de cada repositorio) de una release e iteración concreta.
Entradas:

1) Entorno
2) Release
3) Iteración

Flujo:

1) Consulta en base de datos interna con el entorno, la release y la iteración.
2) Convierte los datos de la consulta en JSON y lo devuelve.

Salida: Archivo JSON con la información solicitada.

RF15: Servicio REST POST instalación

Código Requisito Funcional: RF15.
Nombre: Servicio REST POST instalación.
Requisito de Usuario de partida: RU4.
Descripción: Servicio REST que permite registrar o modificar una instalación.
Entradas: Archivo JSON con los siguientes campos:

1) Entorno
2) Release
3) Iteración
4) Demandas involucradas
5) Descripción (opcional)
6) Changeset de cada repositorio (mínimo el changeset de un repositorio)

Flujo:

1) Comprobar que no existe la iteración en ese entorno y release.
2) Registrar en base de datos interna todas las entradas.
3) Acceder al Excel de IC (en el caso de que el entorno sea IC), crear una nueva columna (u hoja si es una nueva release) y rellenar los datos de la entrada.
4) Guardar el Excel en el sharepoint y devolver el código de respuesta que corresponda.

Salida: Archivo JSON con el código y mensaje de respuesta del servicio.

RF16: Mostrar manual servicios REST

Código Requisito Funcional: RF16.
Nombre: Mostrar manual servicios REST.
Requisito de Usuario de partida: RU8.
Descripción: Muestra el manual de utilización de los servicios REST.
Entradas: Ninguna.
Flujo: Imprimir por pantalla el manual de utilización.
Salida: Una pantalla accesible mediante un enlace que muestre de forma clara como utilizar los servicios REST que ofrece el programa.

RF17: Mostrar logs de las aplicaciones

Código Requisito Funcional: RF17.
Nombre: Mostrar logs de las aplicaciones.
Requisito de Usuario de partida: RU10.
Descripción: Muestra y enlaza la información relacionada con los logs de cada las aplicaciones instaladas en cada entorno.
Entradas: Ninguna.
Flujo: Imprimir por pantalla los enlaces a los logs.
Salida: Muestra por pantalla los enlaces a los logs generados por cada una de las aplicaciones instaladas en los distintos entornos.

3.3 Requisitos no funcionales

A diferencia de los funcionales, los requisitos no funcionales no se refieren a un servicio específico que tengan que proporcionar, sino que se refieren a propiedades emergentes que ha de cumplir el sistema [29]. Se citan a continuación los que requerirá el sistema a implementar.

RNF1: Diseño de la aplicación

Código Requisito No Funcional: RNF1.
Nombre: Diseño de la aplicación.
Descripción: La aplicación deberá tener las fuentes y colores correspondientes con la compañía.

RNF2: Disponibilidad de la aplicación

Código Requisito No Funcional: RNF2.
Nombre: Disponibilidad de la aplicación.
Descripción: La aplicación estará disponible a todos los empleados siempre que se encuentren en la red interna de la compañía.

RNF3: Avisos de tiempos de carga

Código Requisito No Funcional: RNF3.
Nombre: Avisos de tiempos de carga.
Descripción: Si el proceso de consulta de información va a ser considerablemente alto deberá informarse de ello antes de proceder a realizar la petición.

RNF4: Carga de datos fuera del horario de trabajo

Código Requisito No Funcional: RNF4.
Nombre: Carga de datos fuera del horario de trabajo.
Descripción: Siempre que sea posible, la carga de información se hará a través de procesos que no perjudiquen a los trabajadores en horarios, preferiblemente, nocturnos.

RNF5: Usabilidad de la aplicación

Código Requisito No Funcional: RNF5.
Nombre: Usabilidad de la aplicación.
Descripción: La aplicación deberá ser lo más sencilla e intuitiva posible, mostrando menús claros con nombres concretos.

RNF6: Sesiones simultaneas

Código Requisito No Funcional: RNF6.
Nombre: Sesiones simultaneas.
Descripción: La aplicación estará disponible de manera que varios usuarios puedan descargar o actualizar datos de manera simultánea en la medida de lo posible.

RNF7: Mantenimiento de la aplicación

Código Requisito No Funcional: RNF7.
Nombre: Mantenimiento de la aplicación.
Descripción: La aplicación contará con una documentación completa, tanto de la parte técnica como la de usabilidad para que pueda ser modificada y actualizada fácilmente.

RNF8: Portabilidad

Código Requisito No Funcional: RNF8.
Nombre: Portabilidad.
Descripción: La aplicación deberá estar instalada en un sistema orientado a la posibilidad de tener que ser transferido a otra ubicación de la manera más rápida y sencilla posible.

4 DISEÑO Y DESA/RROLLO

En este capítulo se detallará la solución planteada para resolver todos los problemas encontrados y se recopilarán todas las herramientas utilizadas para desarrollar el portal.

4.1 Solución planteada

Este apartado trata sobre el diseño por el que se ha optado para el portal web. Se hablará sobre la arquitectura elegida, el alcance de la aplicación, las características de la base de datos.

4.1.1 Arquitectura

Como se ha venido mencionando a lo largo del documento, se ha optado por la implementación de un portal web siguiendo la infraestructura WAMP, un servidor Apache, con una base de datos MySQL y programado en PHP sobre el sistema operativo Windows [30].

La capa de presentación (front-end) estará desarrollado en PHP y javascript, utilizado la técnica de AJAX en la realización de consultas asíncronas a base de datos [31] para mejorar la experiencia de uso y los tiempos de carga al mostrar la información deseada. Todos los accesos a la base de datos (back-end) se harán mediante la extensión MySQLi de PHP, aunque se necesitará también OCI8 (extensión para interactuar con bases de datos Oracle mediante PHP [32]).

Además del acceso a la base de datos interna y a la de Oracle, se necesitarán también al Sharepoint donde se almacenan los documentos para el departamento de Calidad y al de instalaciones de Integración Continua, estos dos últimos accesos se realizarán conectándolos como unidad de red al servidor donde se alojará el portal.

El servidor tendrá acceso a la herramienta Plastic SCM para realizar las tareas de verificación del formato de ficheros PL/SQL así como averiguar en qué ramas se encuentran las peticiones ingresadas.

Se necesitará implementar una serie de peticiones get para la recopilación de versiones del entorno de preproducción. Al no tener acceso a la base de datos concreta se accederán a éstos por la interfaz web, analizando el código html de la misma.

Por último, el sistema ofrecerá una API REST para que otras herramientas puedan manipular, en formato JSON, información relativa a la línea base e instalaciones de los entornos.

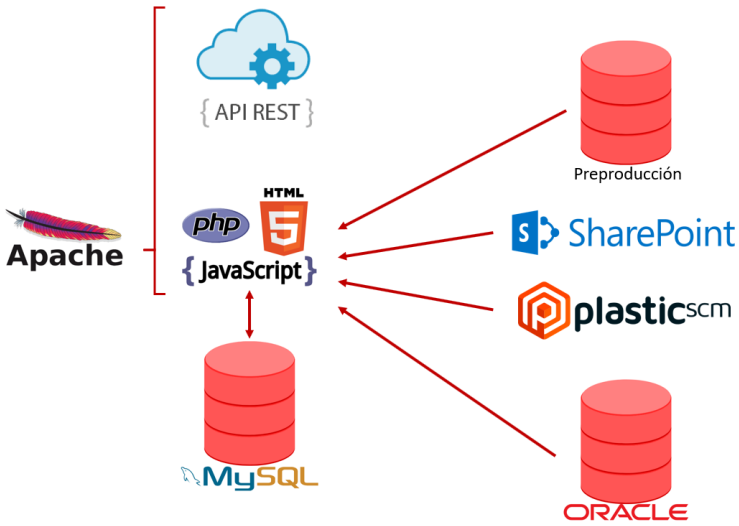


Figura 4.1: Arquitectura del sistema
Fuente: propia

4.1.2 Alcance

Se incluyen en el alcance de la aplicación un plan de proyecto que consta de la metodología empleada, recopilación de requisitos, diseño y definición de las tecnologías utilizadas; creación del portal, su instalación, demostración y manual de usuario; soporte, mantenimiento y evolución.

4.1.3 Base de Datos

Como hemos mencionado en el apartado de la arquitectura, se utilizará un modelo de base de datos relacional MySQL y su diseño será bastante sencillo, utilizando las siguientes tablas:

- **ENTORNOS.** Guardará el nombre de los entornos donde se vayan a promocionar las instalaciones.
- **INSTALACIONES.** Almacenará todos los datos relativos a una instalación concreta: el nombre, la iteración, el entorno, demandas involucradas, una

descripción opcional, el changeset de cada repositorio (al menos uno) y la fecha en la que se da de alta.

- **LB_IC.** Contendrá los changeset de cada uno de los repositorios instalados en el entorno de IC, un identificador entero y la fecha de actualización de la tabla.
- **LB_INT.** Incluirá las versiones de cada uno de los repositorios instalados en el entorno de integración, un identificador entero y la fecha de actualización de la tabla.
- **LB_PRE.** Acumulará las versiones de cada uno de los repositorios instalados en el entorno de preproducción, un identificador entero y la fecha de actualización de la tabla.

4.2 Herramientas y tecnologías para el desarrollo

A continuación se explicarán las herramientas y tecnologías que se han utilizado para realizar la etapa de desarrollo del portal, que será la más larga de todas, sin contar el mantenimiento del portal.

4.2.1 Visual Basic

El departamento de Calidad necesita trabajar con archivos Excel con un tamaño muy grande (en ocasiones de más de 12.000 registros), la manera más sencilla y rápida de automatizar cualquier tarea que involucre este tipo de ficheros es utilizar Visual Basic (VBA), el lenguaje de Microsoft utilizado para la creación de aplicaciones Windows [33].

En nuestro caso se utilizará para crear una serie de macros (nombre que recibe un script en Excel) que se ejecute de manera automática para pasar un filtro y hacer una criba de los registros que no necesitan ser analizados por el departamento. De esta manera se consigue que el encargado de hacerlo de forma manual reciba directamente la información que necesita.

Para otras tareas que involucran ficheros Excel (como la actualización automática del registro de instalaciones o el volcado con un formato concreto de una consulta Oracle) se ha utilizado una librería de PHP externa para facilitar la edición de estos documentos, ya que VBA en algunas ocasiones es demasiado tedioso de utilizar según para que propósitos.

4.2.2 Procesos batch

Todas las actualizaciones de líneas base, generación de documentación, consultas para exportación y filtrado se realizan mediante procesos internos fuera del horario de trabajo para no ralentizar la aplicación y para que los tiempos de carga sean los menores posibles.

A los procesos encargados de la realización de pruebas repetitivas y automáticas con gran cantidad de información en los que el usuario no tiene ninguna interacción con el programa se les conoce como procesos batch (procesamiento por lotes) [34].

En este caso se ha utilizado la herramienta "Programador de tareas" de Windows que permite ejecutar un script .bat cada cierto tiempo de manera automática. En este script habrá una llamada al correspondiente programa php que se ha utilizado para hacer todas las actualizaciones de la base de datos y la generación de ficheros por ser más sencillo de programar.

4.2.3 Servidor y BD

Para gestionar tanto el servidor como la base de datos se ha utilizado la herramienta Xampp, ya que es uno de los paquetes de software más sencillos de instalar y utilizar para desarrollar el portal. Una vez desarrollado, se ha instalado en un entorno WAMP; es decir, se ha levantado un servidor Apache y un sistema de gestión de bases de datos MySQL en Windows utilizando principalmente PHP como lenguaje de programación principal.

Apache es uno de los software de servidor web de código abierto más utilizados actualmente debido a que es multiplataforma (se puede instalar tanto en servidores Unix como Windows), fácil de configurar y con una amplia comunidad por lo que cualquier problema tendrá una rápida solución [35].

MySQL ha sido el gestor de base de datos SQL utilizado para este proyecto. Para facilitar su administración se ha utilizado la interfaz gráfica que ofrece SQLyog en su versión gratuita. Se ha utilizado principalmente esta herramienta para la creación de las tablas y para la comprobación de la carga de los datos, ya que su interfaz permite ver la información de una manera más amigable. Gracias al editor SQL que incluye, se pueden realizar consultas de manera que antes de realizarlas las llamadas desde el código PHP se comprueba su salida en la herramienta.

4.2.4 Swagger

El API REST se encargará de ofrecer una serie de servicios para que otras herramientas manejen información de manera que no tengan que utilizar la interfaz gráfica del portal. Es una forma de automatizar aun más todos los procesos de exportación e importación de datos, en este caso relativos a instalaciones, releases y líneas base de entornos.

Se ha decidido utilizar Swagger como herramienta para el diseño, construcción y documentación del API REST desarrollada, en particular se ha utilizado el editor online que ofrece. Swagger es una serie de reglas, especificaciones y herramientas que nos ayudan a documentar APIs [36].

Aunque esta herramienta ofrece muchas más opciones, como la comprobación del correcto funcionamiento de servicios online o la creación de script de pruebas para

automatizarlas, en este proyecto únicamente se ha creado una documentación visual e interactiva para que a la hora de utilizar el API sea no haya ningún tipo de equivocación.

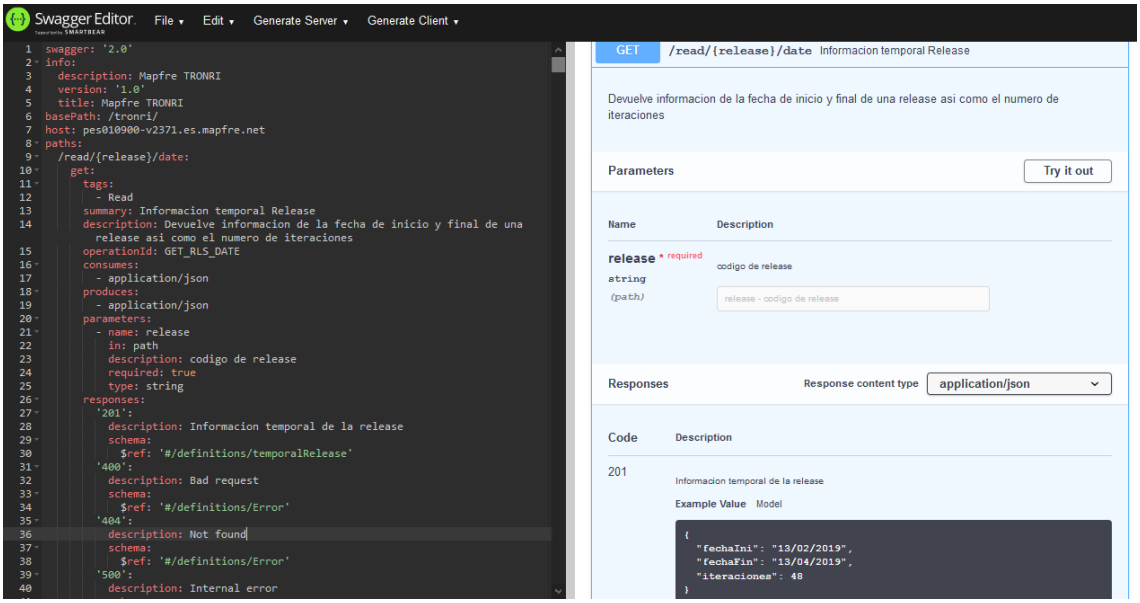


Figura 4.2: Captura interfaz swagger
Fuente: propia

5 PRUEBAS Y USO DEL PORTAL

Este capítulo contendrá todo lo relacionado al siguiente paso al desarrollo, donde se mencionará el tipo de pruebas realizadas, se explicará quienes y de qué manera interactuarán con el portal y cuáles serán las tareas de mantenimiento y soporte.

5.1 Pruebas

El objetivo de este apartado es resumir el proceso que se ha seguido a la hora de validar todos los requisitos implementados en la anterior etapa. Se han realizado pruebas unitarias, de validación y de regresión.

Pruebas unitarias

Consisten en comprobar el correcto funcionamiento de cada uno de los módulos que componen el portal. Se han realizado pruebas que corresponden al correcto formato de introducción de parámetros en los formularios y en el api, mostrando mensajes de error en el caso de introducir caracteres inválidos. Para la realización de estas pruebas se han intentado insertar cadenas de caracteres correspondientes a elementos válidos e inválidos, y comprobando que la detección de inválidos funciona correctamente.

Además, se han realizado pruebas unitarias en el parser de código de manera que se han creado script PL/SQL de pruebas que contengan errores de formato, y así comprobar de una manera rápida mediante un log cuáles son los que detecta y cuáles no. En este aspecto se tuvieron algunos problemas al observar que el instalador daba errores aunque el código mal formateado estuviera en comentarios. Esto se solventó a base de la realización de gran cantidad de pruebas en el script.

Pruebas de validación

Las pruebas de validación sirven para verificar que se cumplen todos los requisitos, tanto funcionales como no funcionales. La realización de estas pruebas consistió en la comprobación de que la información exportada mediante el automatismo del portal era la misma que si se generaba de manera manual. Todas estas pruebas se realizaron de manera manual, sin ningún tipo de automatismo, de manera que supuso bastante pérdida de tiempo en algunos casos, aunque en muchos otros no había ninguna forma de automatizar estas pruebas, como es el ejemplo de los repositorios involucrados en una demanda o la comprobación de si un servidor está caído o no, que se debía hacer de forma visual.

Pruebas de regresión

Como ya hemos hablado anteriormente, el proyecto se compone de módulos que se unen en un solo portal web. Se han tenido que realizar pruebas de regresión cada vez que un nuevo módulo es integrado al portal, de tal manera que al unirlo, los componentes sigan funcionando y no haya ningún problema. Esto se ha realizado mediante nuevas pruebas unitarias sobre los componentes anteriores con el nuevo instalado.

Cabe mencionar que, a pesar de que se comprobaron de manera individual y en conjunto toda la funcionalidad del portal, se detectaron errores de funcionamiento por parte del cliente debido a la falta de tipos de casos de prueba, que se fueron solucionando progresivamente cuando se informaban.

Pruebas con usuarios

Estas pruebas consistieron en dos partes: primero se realizó una demostración de cómo funcionaba el automatismo con un grupo de usuarios que iban a utilizar la herramienta de manera que se explique el proceso a seguir; la segunda consistió en la realización por parte de los usuarios una prueba para asegurar el correcto conocimiento del método de utilización.

5.2 Funcionalidad

En términos generales, habrá un encargado para cada entorno que deberá realizar las integraciones y su cometido será rellenar un formulario con los datos de la instalación. Se proporciona un formulario que obliga al usuario a introducir los datos necesarios (entorno, iteración, release, al menos una demanda y el changeset de un repositorio). Gracias a la API REST proporcionada para realizar este cometido, se puede automatizar este proceso y así no cometer posibles errores humanos al introducir los datos en el formulario.

Antes de hacer una integración se deberá verificar el formato de los archivos PL/SQL y así prever un posible error al instalarlos, de manera que habrá otro grupo de personas que se encarguen de realizar la descarga de las diferencias formateadas desde el portal. Se muestra un formulario para introducir changeset o versiones, la opción de descargarlas con un log de los errores de formato encontrados o añadir una corrección automática.

Los encargados de realizar los cálculos relativos a Calidad tendrán un apartado para descargar toda la documentación necesaria, que se actualizará todas las noches.

A la hora de realizar pruebas de una funcionalidad concreta se deberá acceder al entorno donde esté desplegada la versión que se quiera probar, por lo que cualquiera que tenga que hacer verificaciones podrá consultar esta información en la aplicación, así como llevar el control sobre las ramas involucradas en una demanda.

La aplicación se encarga de realizar un ping a cada servidor de cada entorno para verificar constantemente si están operativos o no, de esta manera, antes de mandar un correo innecesario preguntando por la disponibilidad, el usuario podrá acceder y comprobarlo de una manera sencilla y rápida.

Se adjunta la documentación de la API REST en formato JSON para que con cualquier intérprete swagger se puedan crear herramientas externas para el acceso a la información que esta API ofrece.

Aunque la información a la que accederá cada uno de los equipos será muy distinta, cada uno de ellos podrá acudir a todos los apartados del portal. Al estar migrando a la metodología DevOps se requiere una involucración de todos los equipos sobre las nuevas prácticas y el acceso a la información y comunicación ha de ser mucho más rápida y sencilla [37]. De esta manera se agiliza la transmisión de contenidos entre equipos, evitando correos o desplazamientos innecesarios.

5.3 Mantenimiento

Una vez hecha la instalación del portal, se ha creado un manual de usuario en el que se especifica cada una de las tareas que se pueden realizar mediante la herramienta. Una vez entregado y revisado por el usuario le sigue la etapa de mantenimiento, que consta de cuatro tipos: correctivo, adaptativo, perfectivo y preventivo [38].

Mantenimiento correctivo

Aunque se ha realizado una gran batería de pruebas para comprobar el correcto funcionamiento del portal, es posible que surjan nuevos defectos en el uso del mismo, esto pueden ser salidas inesperadas, la posibilidad de que el sistema se caiga, tiempos de procesamiento demasiado altos, etc.

Mantenimiento adaptativo

Debido a la alta tendencia de actualización del entorno de trabajo, es posible que la herramienta no sea compatible. En la actualidad, se está migrando de la utilización de Plastic a bitbucket, por lo que todas las interacciones que había con esta herramienta deberán de adaptarse al nuevo scm. Además, se quiere adaptar el parser del formato de archivos PL/SQL para que se lance automáticamente al realizar una subida de un entorno a otro, por lo que se tendrá que modificar la forma en la que se hace actualmente.

Mantenimiento perfectivo

Se podrá añadir funcionalidad según el usuario lo precise. Todos los procesos del portal han nacido de querer automatizar determinadas acciones, se permite que los usuarios puedan solicitar una revisión de la forma en la que realizan sus tareas repetitivas para intentar que sean lo menos manual posible para evitar fallos humanos y ahorrar tiempo.

Mantenimiento preventivo

Se llevará un mantenimiento sobre tiempos de carga, de acceso y de actualización de datos de manera que la aplicación se vaya actualizando para que evolucione correctamente y se perfeccione.

6 CONCLUSIONES Y TRABAJO FUTURO

El objetivo de este proyecto era la creación de una herramienta que englobara una serie de automatismos para mejorar el rendimiento de los trabajadores que mantienen y desarrollan un aplicativo de gestión de activos de empresa.

El proyecto ha supuesto todo un reto ya que ha sido el primer proyecto en el que el autor ha trabajado en el ámbito profesional. Se ha podido valorar positivamente todo el apoyo recibido por parte de los usuarios del portal, que a la vez eran compañeros, así como la paciencia que han tenido a la hora de comprender su metodología de trabajo y los procesos que realizaban para analizarlos y automatizarlos.

Se ha podido poner en práctica muchos conocimientos adquiridos a lo largo del Grado en Ingeniería Informática como es la programación, estructuras de datos, análisis y diseño del software y de algoritmos de forma que se han unificado en un mismo proyecto.

A lo largo de la realización del proyecto no se ha dejado de aprender la forma en la que se trabaja en un entorno profesional, sobre todo en la metodología utilizada y el nuevo enfoque con la transición hacia DevOps. También sorprende la cantidad de trabajo que se realiza de manera manual. Siendo nuestro campo la informática, esperaba que hubiera muchos más automatismos.

Durante el desarrollo, se ha afianzado la creencia de que para realizar un buen trabajo hay que hacer una buena planificación, analizando a conciencia los requerimientos y pensando en posibles líneas de desarrollo para que todas funcionen como es debido al ser integradas y analizando también las tecnologías utilizadas por el usuario y su forma de trabajar. Lo mismo pasa con las pruebas realizadas, que han supuesto en muchos casos un ahorro de tiempo considerable, aunque en ocasiones han faltado casos de prueba, lo que supuso una inversión de tiempo considerable solucionar ciertos errores, debido a la experiencia en muchas de las acciones a automatizar.

Se ha aprendido a utilizar herramientas que no había usado nunca como son Plastic SCM, Polarion, Swagger SQL*Plus y se han adquirido muchos conocimientos sobre promociones a entornos, para que se utilizan, en qué proyectos debería haber uno u otro y el futuro del departamento con DevOps, que significará un cambio cultural considerable.

Las futuras líneas de trabajo están orientadas en dos direcciones, en primer lugar, evolucionar el portal de manera que sea lo más autónomo posible actualizando las instalaciones automáticamente y una segunda dirección que sería en relación con añadir más funcionalidad conforme los usuarios las demanden.

Además sería conveniente que, al estar migrando a la metodología de DevOps, ciertos mecanismos como el analizador de código se lanzaran de manera automática, como se ha mencionado anteriormente, al realizar las subidas de código a las ramas de Integración y Preproducción.

Puesto que se sigue dando soporte y mantenimiento al portal, se tiene un contacto directo con el usuario de manera que siguen llegando peticiones para incluir nuevas pestañas o nuevos repositorios que se crean y deben aparecer en líneas base.

REFERENCIAS

[1] OBS Business School, «¿Qué son las metodologías de desarrollo de software?,» [En línea]. Available: https://obsbusiness.school/es/blog-project-management/metodologia-agile/que-son-las-metodologias-de-desarrollo-de-software.

[2] Ryte Wiki, «Modelo en Cascada,» [En línea]. Available: https://es.ryte.com/wiki/Modelo_en_Cascada.

[3] D. Flores, «¿QUÉ ES UN SISTEMA CONTROLADOR DE VERSIONES (SCM)?,» 23 12 2016. [En línea]. Available: https://telumsoft.com/sistema-controlador-versiones-scm/.

[4] mdepedro, «Trabajo distribuido con Plastic,» 3 2008. [En línea]. Available: http://codicesoftware-es.blogspot.com/2008/03/trabajo-distribuido-con-plastic.html.

[5] Anónimo, «Plastic SCM,» [En línea]. Available: https://es.wikipedia.org/wiki/Plastic_SCM.

[6] Plastic SCM, «PLASTIC SCM CLI GUIDE,» [En línea]. Available: https://www.plasticscm.com/documentation/cli/plastic-scm-version-control-cli-guide.

[7] riptutorial, «Empezando con plsql,» [En línea]. Available: https://riptutorial.com/es/plsql.

[8] pc-solucion, «Diferencias entre SQL y PL/SQL,» 19 4 2018. [En línea]. Available: https://pc-solucion.es/2018/04/19/diferencias-entre-sql-y-pl-sql/.

[9] Oracle, «Oracle SQL Developer,» [En línea]. Available: https://www.oracle.com/es/database/technologies/appdev/sql-developer.html.

[10] Oracle, «SQL*Plus® User's Guide and Reference,» [En línea]. Available: https://docs.oracle.com/cd/B19306_01/server.102/b14357/qstart.htm.

[11] Oracle, «SQL*Plus® User's Guide and Reference,» [En línea]. Available: https://docs.oracle.com/cd/E18283_01/server.112/e16604/ch_twelve015.htm.

[12] JoshL, «Stackoverflow,» 22 9 2008. [En línea]. Available: https://stackoverflow.com/questions/118190/how-do-i-ignore-ampersands-in-a-sql-script-running-from-sql-plus.

[13] Oracle, «SQL*Plus® User's Guide and Reference,» [En línea]. Available: https://docs.oracle.com/cd/E11882_01/server.112/e16604/ch_twelve023.htm.

[14] Oracle, «SQL*Plus® User's Guide and Reference,» [En línea]. Available: https://docs.oracle.com/cd/E18283_01/server.112/e16604/ch_twelve004.htm.

[15] Oracle, «SQL*Plus® User's Guide and Reference,» [En línea]. Available: https://docs.oracle.com/database/121/SQPUG/ch_five.htm.

[16] Anónimo, «UTF-8,» [En línea]. Available: https://es.wikipedia.org/wiki/UTF-8.

[17] Oracle, «Descripción general de Unicode,» [En línea]. Available: https://docs.oracle.com/cd/E26921_01/html/E27143/glmgn.html.

[18] Anónimo, «ASCII,» [En línea]. Available: https://es.wikipedia.org/wiki/ASCII.

[19] Microsoft, «SharePoint,» [En línea]. Available: https://products.office.com/es-es/sharepoint/collaboration.

[20] json.org, «Introducción a JSON,» [En línea]. Available: https://www.json.org/json-es.html.

[21] Gustavo, «Bibliotecas de SharePoint como Ubicaciones de Red,» [En línea]. Available: http://www.gavd.net/servers/sharepointv5/spsv5_item.aspx?top=2&itm=1807.

[22] S. Martínez, «Entornos y metodologías de desarrollo en el software de gestión,» 7 12 2014. [En línea]. Available: https://www.mundoerp.com/blog/entornos-y-metodologias-de-desarrollo-software-gestion/.

[23] V. Gilart Iglesias y A. Capella D'alton, «Entornos para el desarrollo de grandes aplicaciones de,» *Departamento de Tecnología Informática y Computación, Universidad de Alicante.*

[24] Microsoft Azure, «¿Qué es DevOps?,» [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-devops/.

[25] J. Ruiz Cristina, «Qué es DevOps (y sobre todo qué no es DevOps),» 2016. [En línea]. Available: https://www.paradigmadigital.com/techbiz/que-es-devops-y-sobre-todo-que-no-es-devops/.

[26] Red Hat, «¿Qué son la integración/distribución continuas (CI/CD)?,» [En línea]. Available: https://www.redhat.com/es/topics/devops/what-is-ci-cd.

[27] Saurabh, «What is Jenkins? | Jenkins For Continuous Integration,» 22 5 2019. [En línea]. Available: https://www.edureka.co/blog/what-is-jenkins/.

[28] Globe Testing, «Pruebas de regresión,» [En línea]. Available: https://www.globetesting.com/pruebas-de-regresion/.

[29] I. Sommerville, Ingeniería del software, Pearson, 2005.

[30] J. A. Martínez Pla, AMPLIACIÓN DE UN PORTAL WORDPRESS PARA EL DISEÑO DE CURSOS BASADOS EN LA METODOLOGIA PBL, Madrid, 2015.

[31] W3School, «PHP - AJAX Introduction,» [En línea]. Available: https://www.w3schools.com/php/php_ajax_intro.asp.

[32] The PHP Group, «OCI8 - Introducción,» [En línea]. Available: https://www.php.net/manual/es/intro.oci8.php.

[33] Microsoft, «Introducción a VBA en Office,» 14 8 2019. [En línea]. Available: https://docs.microsoft.com/es-es/office/vba/library-reference/concepts/getting-started-with-vba-in-office.

[34] Anónimo, «Procesamiento por lotes,» [En línea]. Available: https://es.wikipedia.org/wiki/Procesamiento_por_lotes.

[35] G. B, «¿Qué es Apache? Descripción completa del servidor web Apache,» 1 11 2019. [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-apache/#Como-funciona-el-servidor-web-Apache.

[36] Chakray, «SWAGGER Y SWAGGER UI: ¿QUÉ ES Y POR QUÉ ES IMPRESCINDIBLE PARA TUS APIS?,» 27 1 2017. [En línea]. Available: https://www.chakray.com/es/swagger-y-swagger-ui-por-que-es-imprescindible-para-tus-apis/.

[37] A. Mejía, «Devops, un cambio cultural para lograr la agilidad,» 10 7 2019. [En línea]. Available: https://pfstech.es/devops-un-cambio-cultural-para-lograr-la-agilidad/.

[38] Departamento de informática de la Universidad de Valencia, «Mantenimiento,» 2000. [En línea]. Available: http://informatica.uv.es/iiguia/2000/IPI/material/tema7.pdf.

[39] Selenium, «About Selenium,» [En línea]. Available: https://www.selenium.dev/about/.

A. Manual del programador

El objetivo de este anexo es mostrar algunos ejemplos de los métodos utilizados para la realización de distintas automatizaciones en el proceso de desarrollo.

a. Comando cm para la ejecución de Plastic SCM en CMD

En los siguientes ejemplos se muestra la utilización del comando cm diff y cm find para descargar las diferencias entre dos changeset/versiones y pasar o no el verificador de código y para encontrar las ramas nombradas de la misma manera que una demanda introducida.

cm diff

```
if($radio=="cs"){
  for($i=0; $i<count($aRepositoriosPlastic); $i++){
    if($aCsVer1[$i] and $aCsVer2[$i]){
      exec("cm diff
cs:" . $aCsVer1[$i] . "@" . $aRepositoriosPlastic[$i] . "@plastic:8087
cs:" . $aCsVer2[$i] . "@" . $aRepositoriosPlastic[$i] . "@plastic:8087  --
download=diferencias/diff_" . $time . "/" . $aRepositoriosPlastic[$i], $aOutput[$i]);
      $flag=1;
    } else if($aCsVer1[$i] and !$aCsVer2[$i]) {
      exec("cm diff
cs:" . $aCsVer1[$i] . "@" . $aRepositoriosPlastic[$i] . "@plastic:8087 --
download=diferencias/diff_" . $time . "/" . $aRepositoriosPlastic[$i], $aOutput[$i]);
      $flag=1;
    }
  }
} else if($radio=="version"){
  for($i=0; $i<count($aRepositoriosPlastic); $i++){
    if($aCsVer1[$i] and $aCsVer2[$i]){
      exec("cm diff
lb:" . $aCsVer1[$i] . "@" . $aRepositoriosPlastic[$i] . "@plastic:8087
lb:" . $aCsVer2[$i] . "@" . $aRepositoriosPlastic[$i] . "@plastic:8087  --
download=diferencias/diff_" . $time . "/" . $aRepositoriosPlastic[$i], $aOutput[$i]);
      $flag=1;
    } else if($aCsVer1[$i] and !$aCsVer2[$i]) {
      exec("cm diff
cs:" . $aCsVer1[$i] . "@" . $aRepositoriosPlastic[$i] . "@plastic:8087 --
download=diferencias/diff_" . $time . "/" . $aRepositoriosPlastic[$i], $aOutput[$i]);
      $flag=1;
    }
  }
}

if($parser["p"]) {
  for($i=0; $i<count($aRepositoriosPlastic); $i++){
    if($aCsVer1[$i]){
      exec("php script/parser.php
WorkspaceDir=diferencias/diff_" . $time . "/" . $aRepositoriosPlastic[$i] . "
LogFile=diferencias/diff_" . $time . "/" . $aRepositoriosPlastic[$i] . "/log.txt");
      $flagParser=1;
```

```
    }
  }
} else if($parser["a"]) {
  for($i=0; $i<count($aRepositoriosPlastic); $i++){
    if($aCsVer1[$i]){
      exec("php script/parser.php -a
WorkspaceDir=diferencias/diff_" . $time . "/" . $aRepositoriosPlastic[$i] . "
LogFile=diferencias/diff_" . $time . "/" . $aRepositoriosPlastic[$i] . "/log.txt");
      $flagParser=1;
    }
  }
}

if($flag) {
  $zipDif = "diferencias/diff_" . $time . ".zip";
  exec('"C:\Program Files\\7-Zip\\7z.exe" a ' . $zipDif . '
diferencias/diff_' . $time);
}
```

cm find

```
$query = "cm find branches where name like '" . $demanda . "' on repositories ";

for ($cont = 0; $cont < count ($aRepositoriosPlastic); $cont++)
  $query = $query . "'" . $aRepositoriosPlastic[$cont] . "',";

$query = rtrim ($query, ",") . "--nototal";
$aDatos = explode ("\n", shell_exec ($query));
$fecha = date ("Y-m-d-H-i-s");
$ficheroQuery = "temp/query_" . $fecha . ".txt";
$fp = fopen ($ficheroQuery, 'w');
fwrite ($fp, $query);
fclose ($fp);
```

b. Tratamiento de archivos Excel mediante php

Se adjuntan una serie de ejemplos de tratamiento de ficheros Excel para realizar la criba automática de lo que anteriormente se realizaba de manera manual.

Creación de un Excel dada una consulta SQL

```
function crearExcelAgoraIC($resultadoSQL) {
  $plantillaExcel = "dashBoard/_AgoraIC.xlsx";
  $excelEntrega = "dashBoard/descarga/AgoraIC.xlsx";
  copy ($plantillaExcel, $excelEntrega);

  $spreadsheet = \PhpOffice\PhpSpreadsheet\IOFactory::load($excelEntrega);
  $sheet = $spreadsheet->getSheetByName("Exportar Hoja de Trabajo");

  for($row=0; $row<count($resultadoSQL); $row++){
    $aRes = explode("###", $resultadoSQL[$row]);
    for($column=0; $column<22; $column++){
      $sheet->setCellValueByColumnAndRow($column+1, $row+2, $aRes[$column]);
    }
  }
```

```
  $writer = new Xlsx($spreadsheet);
  $writer->save($excelEntrega);
}
```

Acceso y formateado de archivos Excel de Sharepoint Clarity

```
function exportExcelClarity() {
  $plantillaExcelAbiertas = "Y:\Datos\data_in\peticiones\p_tron_abiertas.xlsx";
  $plantillaExcelCerradas = "Y:\Datos\data_in\peticiones\p_tron_cerradas.xlsx";
  $excelTemporal = "temp/temporal.xlsx";
  $excelEntrega = "dashBoard/descarga/Demandas CLARITY.xlsx";
  $excelBase = "dashBoard/_Demandas CLARITY.xlsx";

  $export = formateaSalida($plantillaExcelAbiertas, $excelTemporal, 0);
  $export2 = formateaSalida($plantillaExcelCerradas, $excelTemporal, 1);

  copy ($excelBase, $excelEntrega);
  $spreadsheet = \PhpOffice\PhpSpreadsheet\IOFactory::load($excelEntrega);
  $sheet = $spreadsheet->getSheetByName("Lista de incidentes1");

  for($row=0; $row<count($export); $row++){
    $filaNew = $export[$row];
    for($col=0; $col<count($filaNew); $col++){
      $sheet->setCellValueByColumnAndRow($col+1, $row+1, $filaNew[$col]);
    }
  }

  for($row2=0; $row2<count($export2); $row2++){
    $filaNew = $export2[$row2];
    for($col=0; $col<count($filaNew); $col++){
      $sheet->setCellValueByColumnAndRow($col+1, $row+1, $filaNew[$col]);
    }
    $row++;
  }

  $writer = new Xlsx($spreadsheet);
  $writer->save($excelEntrega);
}

function formateaSalida($plantillaExcel, $excelTemporal, $flag) {
  $export = $aIndexColumn = $cabeceraSalida = array();
  $cabeceraSalida[] = "Código Petición";
  $cabeceraSalida[] = "Descripción de la petición";
  $cabeceraSalida[] = "Versión";
  $cabeceraSalida[] = "Analista Oferta";
  $cabeceraSalida[] = "Analista";
  $cabeceraSalida[] = "Actividad Actual";
  $cabeceraSalida[] = "Archivos Adjuntos";
  $cabeceraSalida[] = "Fecha de creación";
  $cabeceraSalida[] = "Última Actualización";
  $cabeceraSalida[] = "Propietario";
  $cabeceraSalida[] = "Aplicación";
  $cabeceraSalida[] = "Solicitante";
  $cabeceraSalida[] = "Asignado";
  $cabeceraSalida[] = "Tipo comunicación indicada";
  $cabeceraSalida[] = "Fecha comunicación";
```

```
  $cabeceraSalida[] = "Fecha necesidad";
  $cabeceraSalida[] = "Nº de veces que se entrega";
  $cabeceraSalida[] = "Nº de veces en construcción";
  $cabeceraSalida[] = "Nº de veces en análisis";
  $cabeceraSalida[] = "Nº de veces en comunicada";
  $cabeceraSalida[] = "Nº de veces en espera especificaciones";
  $cabeceraSalida[] = "Nº de veces en implatación";
  $cabeceraSalida[] = "Nº de veces en priorización ejecución";
  $cabeceraSalida[] = "Nº de veces en pruebas funcionales";
  $cabeceraSalida[] = "U.T.E.S. Estimadas";

  copy ($plantillaExcel, $excelTemporal);

  $spreadsheet = \PhpOffice\PhpSpreadsheet\IOFactory::load($excelTemporal);
  $sheet = $spreadsheet->getSheetByName("Lista de incidentes1");
  $numRows = $sheet->getHighestRow();
  $maxColumn = $sheet->getHighestColumn();
  $numColumns = \PhpOffice\PhpSpreadsheet\Cell\Coordinate::columnIndexFromString($
maxColumn);

  for($i=1; $i<$numColumns+1; $i++){
    if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[0]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[1]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[2]) {$v=$i; $aIndexColumn[]=$i;}
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[3]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[4]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[5]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[6]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[7]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[8]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[9]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[10]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[11]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[12]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[13]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[14]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[15]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[16]) $aIndexColumn[]=$i;
```

```
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[17]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[18]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[19]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[20]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[21]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[22]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[23]) $aIndexColumn[]=$i;
    else if($sheet->getCellByColumnAndRow($i, 1)-
>getValue() == $cabeceraSalida[24]) $aIndexColumn[]=$i;
  }

  if($flag==0){
    $cabecera = array();
    foreach($aIndexColumn as $column){
      $cabecera[] = $sheet->getCellByColumnAndRow($column, 1)->getValue();
    }
    $export[] = $cabecera;
  }

  for($j=2; $j<$numRows+1; $j++){
    $version = $sheet->getCellByColumnAndRow($v, $j)->getValue();
    if(strpos($version, "RLS20")!==false){
      $fila = array();
      foreach($aIndexColumn as $column){
        $fila[] = $sheet->getCellByColumnAndRow($column, $j)->getValue();
      }
      $export[] = $fila;
    }
  }
  return $export;
}
```

c. Obtención de la línea base del entorno de Preproducción

Se adjunta un pequeño ejemplo del código sin credenciales ni recursos para mostrar el método de obtención de la línea base del entorno de Preproducción, la cual solo se tiene acceso vía web.

```
function recuperaPre() {
  $user = "";
  $pass = "";
  $uri = "";
  $entorno = "PRE";
  $aUrl = array ();

  $aUrl[] = file_get_html ('http://' . $user . ':' . $pass . '@' . $uri . 'CI=MI13
```

```
8331&CINAME=PLS_ES_GBBS_NWT_BO-' . $entorno);
  $aUrl[] = file_get_html ('http://' . $user . ':' . $pass . '@' . $uri . 'CI=MI13
8334&CINAME=PLS_ES_GBBS_NWT_FW-' . $entorno);
  $aUrl[] = file_get_html ('http://' . $user . ':' . $pass . '@' . $uri . 'CI=G002
5754&CINAME=PLS_ES_GBBS_TRON_SELENIUM_CORE-' . $entorno);
  $aUrl[] = file_get_html ('http://' . $user . ':' . $pass . '@' . $uri . 'CI=MI19
6077&CINAME=PLS_ES_GBBS_TRONWEB_MD_CORE-' . $entorno);

  $pre = array();
  for ($i = 0; $i < count ($aUrl); $i++) {
      $primeraParte = recuperaVersion ($aUrl[$i]);
      $segundaParte = explode(" ", $primeraParte);
      $pre[$i] = $segundaParte[3];
  }
  return $pre;
}

function recuperaVersion ($proyecto) {
  $cont = 1;
  foreach ($proyecto->find ('td') as $elemento) {
      if ($cont == 12) $valor = trim ($elemento->outertext);
      if ($cont == 20) {
          $valor = trim ($elemento->outertext) . "        ###     " . $valor;
          break;
      }
      $cont++;
  }
  return $valor;
}
```

d. Export de la estructura de la BD MySQL

Se muestra la estructura de la base de datos interna con los nombres de los repositorios para cada entorno que se controla con el portal desarrollado.

```
CREATE TABLE `entornos` (
  `ENTORNO` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`ENTORNO`)
);

CREATE TABLE `instalaciones` (
  `NOMBRE_ENTREGA` VARCHAR(30) NOT NULL,
  `ITERACION` INT(10) NOT NULL,
  `ENTORNO` VARCHAR(30) NOT NULL,
  `DEMANDAS` VARCHAR(500) NOT NULL,
  `DESCRIPCION` VARCHAR(500) DEFAULT NULL,
  `NWT_BO` INT(10) DEFAULT NULL,
  `NWT_FW` INT(10) DEFAULT NULL,
  `NWT_BE` INT(10) DEFAULT NULL,
  `NWT_FE` INT(10) DEFAULT NULL,
  `NWT_CIN` INT(10) DEFAULT NULL,
  `NWT_CMN` INT(10) DEFAULT NULL,
  `NWT_ISU` INT(10) DEFAULT NULL,
  `NWT_LSS` INT(10) DEFAULT NULL,
```

```
  `NWT_NFU` INT(10) DEFAULT NULL,
  `NWT_RNS` INT(10) DEFAULT NULL,
  `NWT_THP` INT(10) DEFAULT NULL,
  `NWT_TST` INT(10) DEFAULT NULL,
  `NWT_TSY` INT(10) DEFAULT NULL,
  `TRN_BPM` INT(10) DEFAULT NULL,
  `TRN_CA` INT(10) DEFAULT NULL,
  `TRN_CB` INT(10) DEFAULT NULL,
  `TRN_CE` INT(10) DEFAULT NULL,
  `TRN_CG` INT(10) DEFAULT NULL,
  `TRN_CO` INT(10) DEFAULT NULL,
  `TRN_DC` INT(10) DEFAULT NULL,
  `TRN_EA` INT(10) DEFAULT NULL,
  `TRN_EM` INT(10) DEFAULT NULL,
  `TRN_EV` INT(10) DEFAULT NULL,
  `TRN_GC` INT(10) DEFAULT NULL,
  `TRN_JB` INT(10) DEFAULT NULL,
  `TRN_RA` INT(10) DEFAULT NULL,
  `TRN_SS` INT(10) DEFAULT NULL,
  `TRN_TRN` INT(10) DEFAULT NULL,
  `TRN_TS` INT(10) DEFAULT NULL,
  `TRN_TST` INT(10) DEFAULT NULL,
  `TRN_TWTST` INT(10) DEFAULT NULL,
  `TRN_TWCLI` INT(10) DEFAULT NULL,
  `TRN_TWCLITST` INT(10) DEFAULT NULL,
  `TRN_TWFW` INT(10) DEFAULT NULL,
  `TRN_TWSERVER` INT(10) DEFAULT NULL,
  `TRN_WTW` INT(10) DEFAULT NULL,
  `TRN_WTWREP` INT(10) DEFAULT NULL,
  `TRN_WTWTST` INT(10) DEFAULT NULL,
  `TRN_WTWBB` INT(10) DEFAULT NULL,
  `TRN_WTWCLI` INT(10) DEFAULT NULL,
  `TRN_WTWCLITST` INT(10) DEFAULT NULL,
  `TRN_WTWFW` INT(10) DEFAULT NULL,
  `TRN_MD` INT(10) DEFAULT NULL,
  `NWT_TRW` INT(10) DEFAULT NULL,
  `TRN` INT(10) DEFAULT NULL,
  `NWT_BOM` INT(10) DEFAULT NULL,
  `NWT_RPT` INT(10) DEFAULT NULL,
  `ISU_API` INT(10) DEFAULT NULL,
  `THP_API` INT(10) DEFAULT NULL,
  `TSY_API` INT(10) DEFAULT NULL,
  `TRN_VIDA` INT(10) DEFAULT NULL,
  `TRS_SELENIUM` INT(10) DEFAULT NULL,
  `FECHA` datetime NOT NULL,
  PRIMARY KEY (`NOMBRE_ENTREGA`,`ITERACION`,`ENTORNO`)
);

CREATE TABLE `lb_integracion` (
  `ID` INT(50) NOT NULL,
  `NWT_BO` VARCHAR(50) DEFAULT NULL,
  `NWT_FW` VARCHAR(50) DEFAULT NULL,
  `NWT_BE` VARCHAR(50) DEFAULT NULL,
  `NWT_FE` VARCHAR(50) DEFAULT NULL,
  `NWT_CMN` VARCHAR(50) DEFAULT NULL,
```

```
  `NWT_CIN` VARCHAR(50) DEFAULT NULL,
  `NWT_ISU` VARCHAR(50) DEFAULT NULL,
  `NWT_RNS` VARCHAR(50) DEFAULT NULL,
  `NWT_LSS` VARCHAR(50) DEFAULT NULL,
  `NWT_THP` VARCHAR(50) DEFAULT NULL,
  `NWT_TSY` VARCHAR(50) DEFAULT NULL,
  `NWT_NFU` VARCHAR(50) DEFAULT NULL,
  `NWT_TRW` VARCHAR(50) DEFAULT NULL,
  `NWT_TST` VARCHAR(50) DEFAULT NULL,
  `TRW` VARCHAR(50) DEFAULT NULL,
  `BPM_CORE` VARCHAR(50) DEFAULT NULL,
  `CA_CORE` VARCHAR(50) DEFAULT NULL,
  `CB_CORE` VARCHAR(50) DEFAULT NULL,
  `CE_CORE` VARCHAR(50) DEFAULT NULL,
  `CG_CORE` VARCHAR(50) DEFAULT NULL,
  `CO_CORE` VARCHAR(50) DEFAULT NULL,
  `DC_CORE` VARCHAR(50) DEFAULT NULL,
  `EA_CORE` VARCHAR(50) DEFAULT NULL,
  `EM_CORE` VARCHAR(50) DEFAULT NULL,
  `EV_CORE` VARCHAR(50) DEFAULT NULL,
  `GC_CORE` VARCHAR(50) DEFAULT NULL,
  `JB_CORE` VARCHAR(50) DEFAULT NULL,
  `MD_CORE` VARCHAR(50) DEFAULT NULL,
  `RA_CORE` VARCHAR(50) DEFAULT NULL,
  `SS_CORE` VARCHAR(50) DEFAULT NULL,
  `TRN_CORE` VARCHAR(50) DEFAULT NULL,
  `TS_CORE` VARCHAR(50) DEFAULT NULL,
  `TST` VARCHAR(50) DEFAULT NULL,
  `TW_TST` VARCHAR(50) DEFAULT NULL,
  `TWCLIENT_CORE` VARCHAR(50) DEFAULT NULL,
  `TWCLIENT_TST` VARCHAR(50) DEFAULT NULL,
  `TWFRAMEWORK` VARCHAR(50) DEFAULT NULL,
  `TWSERVER` VARCHAR(50) DEFAULT NULL,
  `WTW_CORE` VARCHAR(50) DEFAULT NULL,
  `WTW_REPORTS` VARCHAR(50) DEFAULT NULL,
  `WTW_TST` VARCHAR(50) DEFAULT NULL,
  `WTWBACKBASE` VARCHAR(50) DEFAULT NULL,
  `WTWCLIENT_CORE` VARCHAR(50) DEFAULT NULL,
  `WTWCLIENT_TST` VARCHAR(50) DEFAULT NULL,
  `WTWFRAMEWORK` VARCHAR(50) DEFAULT NULL,
  `ISU_API` VARCHAR(50) DEFAULT NULL,
  `THP_API` VARCHAR(50) DEFAULT NULL,
  `TSY_API` VARCHAR(50) DEFAULT NULL,
  `LSS_API` VARCHAR(50) DEFAULT NULL,
  `CMN_API` VARCHAR(50) DEFAULT NULL,
  `FECHA` datetime NOT NULL,
  PRIMARY KEY (`NOMBRE_ENTREGA`,`ITERACION`,`ENTORNO`)
);

 CREATE TABLE `lb_ic` (
  `ID` INT(50) NOT NULL,
  `NWT_BO_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_FW_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_BE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_FE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
```

```
  `NWT_CMN_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_CIN_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_ISU_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_RNS_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_LSS_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_THP_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_TSY_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_NFU_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_TRW_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `NWT_TST_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TRW_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `BPM_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `CA_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `CB_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `CE_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `CG_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `CO_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `DC_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `EA_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `EM_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `EV_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `GC_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `JB_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `MD_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `RA_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `SS_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TRN_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TS_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TST_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TW_TST_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TWCLIENT_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TWCLIENT_TST_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TWFRAMEWORK_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TWSERVER_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `WTW_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `WTW_REPORTS_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `WTW_TST_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `WTWBACKBASE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `WTWCLIENT_CORE_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `WTWCLIENT_TST_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `WTWFRAMEWORK_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `ISU_API_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `THP_API_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `TSY_API_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `LSS_API_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `CMN_API_SNAPSHOT` VARCHAR(50) DEFAULT NULL,
  `FECHA` datetime(6) DEFAULT NULL,
  PRIMARY KEY (`ID`)
);

CREATE TABLE `lb_pre` (
  `ID` INT(10) NOT NULL,
  `ES_GBBS_NWT_BO` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_FW` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_BE` VARCHAR(70) DEFAULT NULL,
```

```
  `ES_GBBS_NWT_FE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_CMN_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_CIN_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_ISU_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_RNS_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_LSS_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_THP_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_TSY_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_NFU_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_TRW_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_TST_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_CORE_BPM` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_CA_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_CB_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_CE_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_CG_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_CO_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_DC_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_EA_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_EM_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_EV_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_GC_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_JB_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_CORE_MD` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_RA_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_SS_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_TRN_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_TS_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_TST` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_TW_TST` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_TWCLIENT_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_TWCLIENT_TST` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_TWFRAMEWORK` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_TWSERVER` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_WTW_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_WTW_REPORTS` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_WTW_TST` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_WTWBACKBASE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_WTWCLIENT_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_WTWCLIENT_TST` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_WTWFRAMEWORK` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_BATCH_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_UTILS_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRONWEB_VIDA_UNIT_LINKED_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_TRON_SELENIUM_CORE` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_ISU_API` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_THP_API` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_TSY_API` VARCHAR(70) DEFAULT NULL,
  `ES_GBBS_NWT_RPT_CORE` VARCHAR(70) DEFAULT NULL,
  `FECHA` datetime(6) DEFAULT NULL,
  PRIMARY KEY (`ID`)
);
```

e. Macro para tratamiento de Excel en VBA

Se adjunta una captura de un ejemplo de macro utilizada para la criba de un fichero Excel concreto al ser considerablemente más rápido de esta manera que realizándolo en PHP.

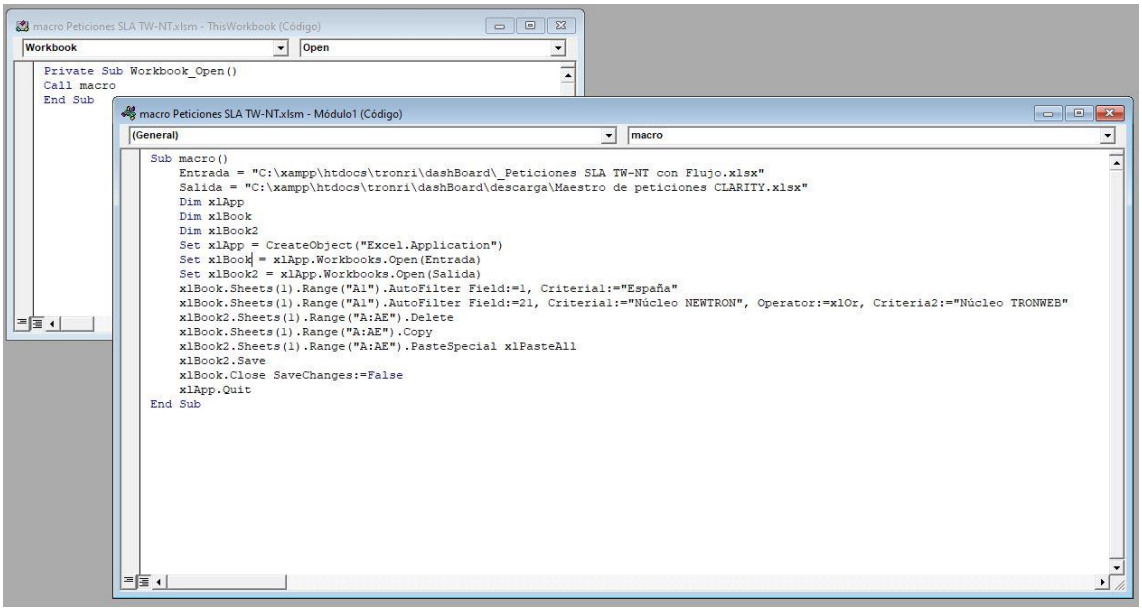
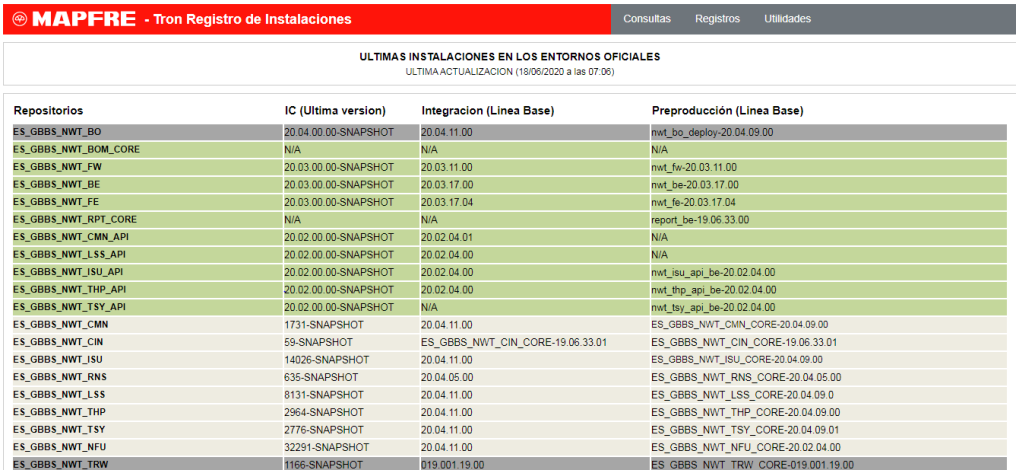


Figura A.1: Captura macro criba Excel
Fuente: propia

f. Capturas de la interfaz del portal desarrollado

Se adjuntan capturas del portal desarrollado comentando a qué parte del producto corresponde.

La primera imagen corresponde a la pantalla de inicio, donde se especifica en que versión o changeset está el aplicativo en los entornos de IC, Integración y Preproducción:



Repositorios	IC (Ultima version)	Integracion (Linea Base)	Preproducción (Linea Base)
ES_GBBS_NWT_BO	20.04.00.00-SNAPSHOT	20.04.11.00	nwt_bo_deploy-20.04.09.00
ES_GBBS_NWT_BOM_CORE	N/A	N/A	N/A
ES_GBBS_NWT_FW	20.03.00.00-SNAPSHOT	20.03.11.00	nwt_fw-20.03.11.00
ES_GBBS_NWT_BE	20.03.00.00-SNAPSHOT	20.03.17.00	nwt_be-20.03.17.00
ES_GBBS_NWT_FE	20.03.00.00-SNAPSHOT	20.03.17.04	nwt_fe-20.03.17.04
ES_GBBS_NWT_RPT_CORE	N/A	N/A	report_be-19.06.33.00
ES_GBBS_NWT_CMN_API	20.02.00.00-SNAPSHOT	20.02.04.01	N/A
ES_GBBS_NWT_LSS_API	20.02.00.00-SNAPSHOT	20.02.04.00	N/A
ES_GBBS_NWT_ISU_API	20.02.00.00-SNAPSHOT	20.02.04.00	nwt_isu_api_be-20.02.04.00
ES_GBBS_NWT_THP_API	20.02.00.00-SNAPSHOT	20.02.04.00	nwt_thp_api_be-20.02.04.00
ES_GBBS_NWT_TSY_API	20.02.00.00-SNAPSHOT	N/A	nwt_tsy_api_be-20.02.04.00
ES_GBBS_NWT_CMN	1731-SNAPSHOT	20.04.11.00	ES_GBBS_NWT_CMN_CORE-20.04.09.00
ES_GBBS_NWT_CIN	59-SNAPSHOT	ES_GBBS_NWT_CIN_CORE-19.06.33.01	ES_GBBS_NWT_CIN_CORE-19.06.33.01
ES_GBBS_NWT_ISU	14026-SNAPSHOT	20.04.11.00	ES_GBBS_NWT_ISU_CORE-20.04.09.00
ES_GBBS_NWT_RNS	635-SNAPSHOT	20.04.05.00	ES_GBBS_NWT_RNS_CORE-20.04.05.00
ES_GBBS_NWT_LSS	8131-SNAPSHOT	20.04.11.00	ES_GBBS_NWT_LSS_CORE-20.04.09.0
ES_GBBS_NWT_THP	2964-SNAPSHOT	20.04.11.00	ES_GBBS_NWT_THP_CORE-20.04.09.00
ES_GBBS_NWT_TSY	2776-SNAPSHOT	20.04.11.00	ES_GBBS_NWT_TSY_CORE-20.04.09.01
ES_GBBS_NWT_NFU	32291-SNAPSHOT	20.04.11.00	ES_GBBS_NWT_NFU_CORE-20.02.04.00
ES_GBBS_NWT_TRW	1166-SNAPSHOT	019.001.19.00	ES_GBBS_NWT_TRW_CORE-019.001.19.00

Figura A.2: Captura index portal
Fuente: propia

En la siguiente imagen se observa el pequeño formulario para consultar una instalación concreta o la línea base de un entorno (el resultado de la consulta será la segunda imagen):

Figura A.3: Captura consultar instalación
Fuente: propia

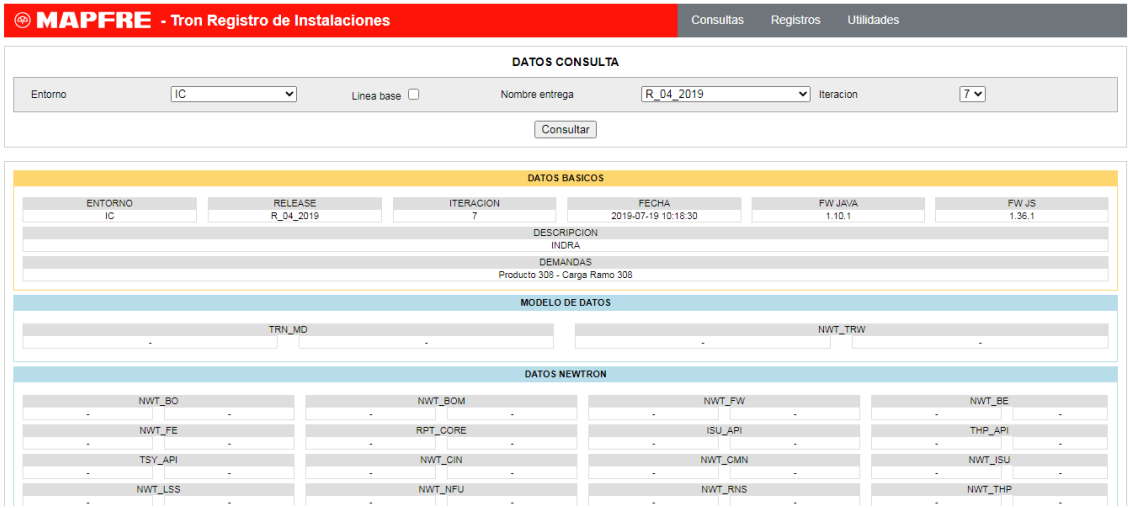


Figura A.4: Captura resultado consultar instalación
Fuente: propia

La siguiente captura corresponde al formulario para el registro de una instalación concreta:

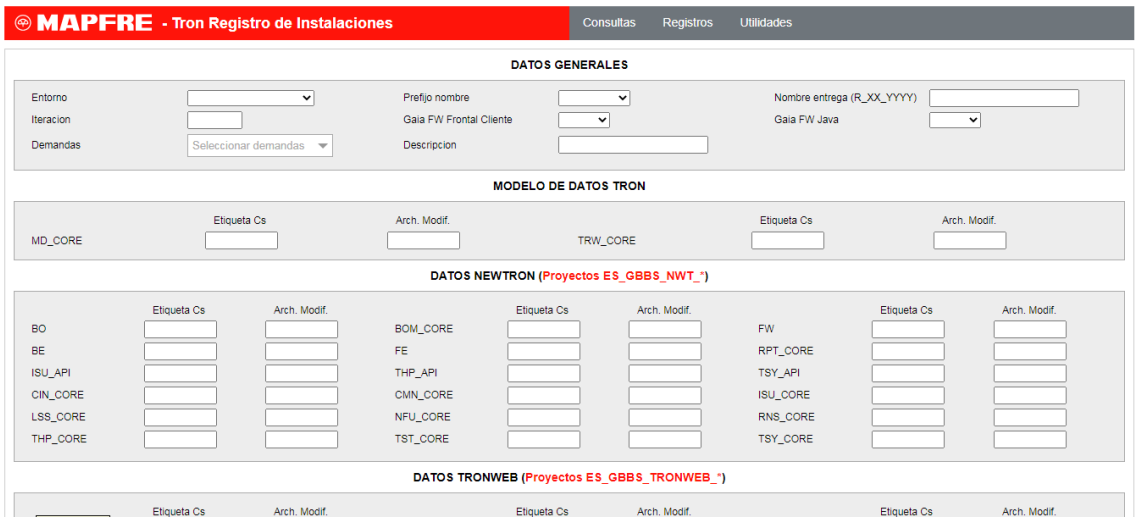


Figura A.5: Captura registrar instalación
Fuente: propia

La siguiente imagen es muy similar a la de consulta, sirve para realizar una modificación de una instalación, primero se elige la iteración (primera imagen) y después se modifican los valores erróneos (segunda imagen):

Figura A.6: Captura modificar instalación
Fuente: propia

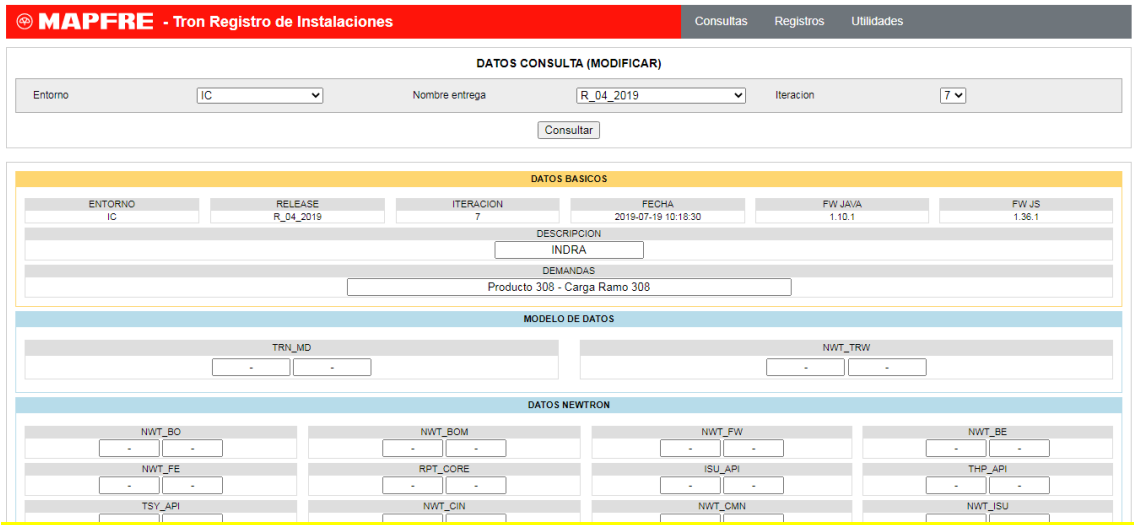


Figura A.7: Captura resultado modificar instalación
Fuente: propia

Introducción de una demanda (primera imagen) para averiguar las ramas donde se encuentra (segunda imagen):

Figura A.8: Captura ramas de una demanda
Fuente: propia

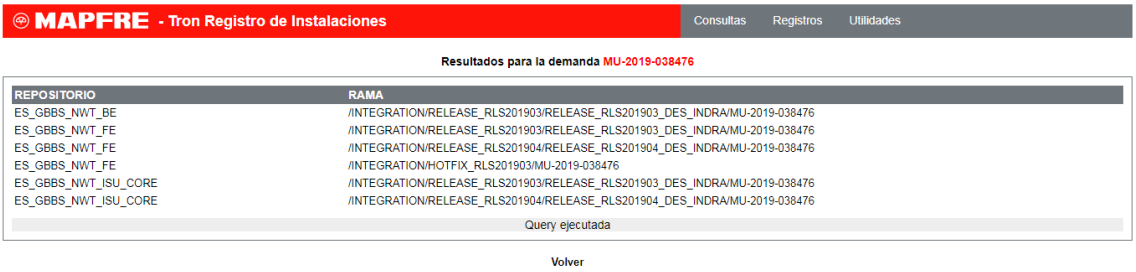


Figura A.9: Captura resultado ramas de una demanda
Fuente: propia

Información sobre las direcciones de los servidores y la disponibilidad de cada uno de estos:

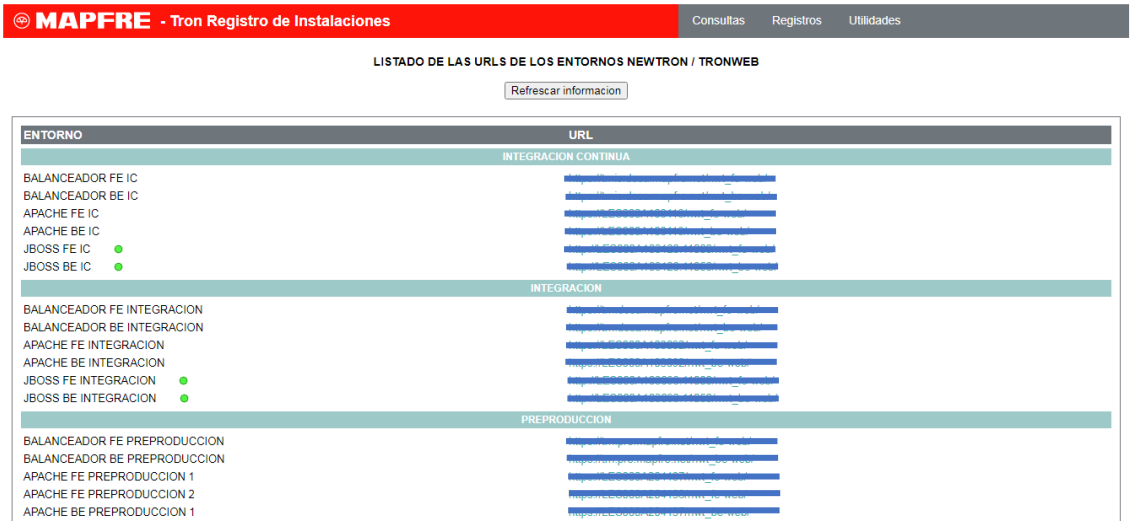


Figura A.10: Captura direcciones entornos
Fuente: propia

Pantalla destinada a la descarga de diferencias con la posibilidad de realizar el analizador sintáctico para el formato de archivos PL/SQL, dando como resultado un zip con los archivos deseados:

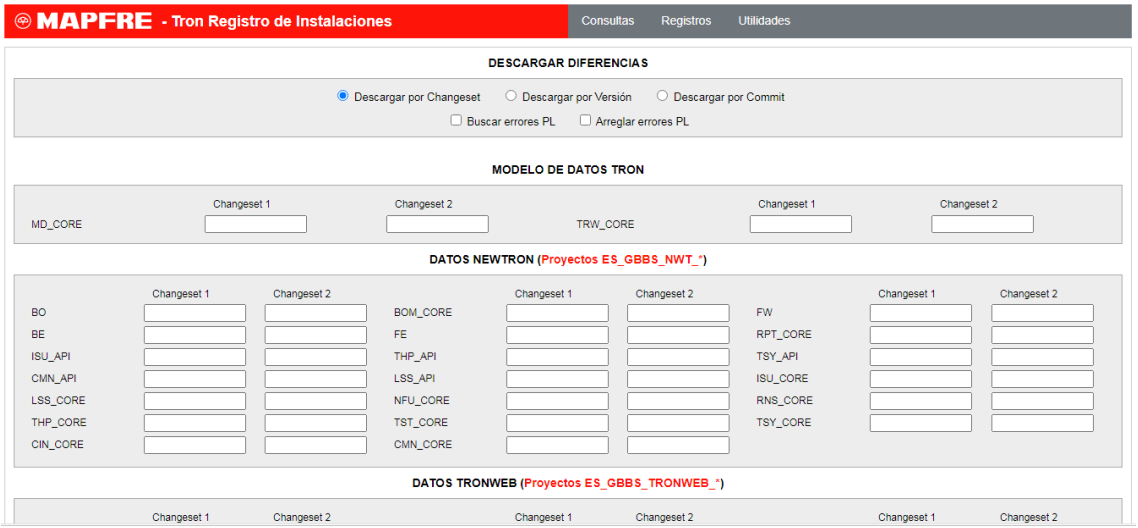


Figura A.11: Captura descarga y analizador de diferencias
Fuente: propia

Apartado de descarga de archivos para la realización de informes de calidad:

Figura A.12: Captura descargas documentación de calidad
Fuente: propia